EFFICIENT LLM ADVERSARIAL TRAINING VIA LOW-RANK DEFENSE AND CIRCUIT-GUIDED SURROGATES

Anonymous authors
Paper under double-blind review

ABSTRACT

Adversarial training is one of the most effective defenses against adversarial attacks, yet the computational cost remains prohibitive at modern scales, especially for large language models (LLMs). While existing mitigation strategies, e.g., latent adversarial training (LAT), have been developed, they still incur a high computational cost. In this work, focusing on LLM-based classifiers, we investigate computation-efficient strategies to speed up LAT from two complementary perspectives: (1) Defense-side optimization: We explore representation fine-tuning (ReFT) within LAT, and reveal a potential issue if there is a mismatch on which tokens to apply ReFT and the attack. (2) Attack-side optimization: When computing adversarial attacks in each LAT iteration, we extract only the relevant circuits from the LLM to construct a lightweight surrogate model, avoiding the computation in the forward-backward passes through the full model during the attack generation. For both perspectives, we provide theoretical justifications and numerical evidence to illustrate the effectiveness of the proposed strategies. Compared to standard LAT with full fine-tuning, our method on average reduces per-step adversarial-training FLOPs by 29.6% while requiring only 0.0118% trainable parameters, at a moderate cost in robustness.

1 INTRODUCTION

Large language models (LLMs) have achieved remarkable performance across diverse tasks, yet remain critically vulnerable to adversarial attacks that manipulate model outputs via crafted prompts (Zou et al., 2023b; Yu et al., 2023; Andriushchenko et al., 2024). These vulnerabilities pose serious risks for real-world deployments, ranging from security-critical classifiers for tasks such as content moderation and filtering to agentic systems (Inan et al., 2023; Howe et al., 2024), making robust defense mechanisms essential. For example, recent studies have shown that small input perturbations can alter the decisions of widely deployed safety classifiers (Achara & Chhabra, 2025; Hackett et al., 2025), while prompt injection has been exploited to induce unsafe downstream actions in AI coding agents (Hart, 2026).

To ensure the robustness of LLMs, one of the most effective defenses is adversarial training (Xing et al., 2021; Zhao et al., 2024), which trains LLMs on adversarial examples to help them better defend against adversarial inputs. However, scaling adversarial training to modern LLMs is computationally infeasible. The cost arises from two sources: (i) updating a large number of model parameters during training, and (ii) repeatedly running the full model to generate adversarial examples via iterative optimization in the token space (Zou et al., 2023b; Howe et al., 2024). Recent works have proposed latent adversarial training (LAT) (Xhonneux et al., 2024; Sheshadri et al., 2024; Casper et al., 2024), which perturbs continuous hidden representations rather than optimizing directly in the discrete token space. This makes the optimization easier than token-level adversarial example generation (Howe et al., 2024), but the computational bottleneck remains significant: running the inner attack with Projected Gradient Descent (PGD) (Madry et al., 2017) for 8 steps alone accounts for approximately two-thirds of the computation in LAT under the Kaplan FLOPs estimate (Kaplan et al., 2020).

While parameter-efficient training strategies such as LoRA and representation fine-tuning (ReFT) (Wu et al., 2024; Ren et al., 2025; Zeng et al., 2025) have been widely adopted, significant gaps persist when applying them in the context of LAT. **(G1)** On the defense side (i.e., updating model parameters to enhance robustness), ReFT requires substantially fewer trainable parameters than

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

LoRA, making it an attractive candidate for efficient adversarial training. However, standard ReFT often applies interventions on task-relevant positions, sometimes only the last token at a specific hidden layer (usually not the final layer). Since adversarial attacks can perturb multiple tokens, it remains unclear whether such position-limited interventions can effectively defend multi-token attacks. **(G2)** On the attack side, while ReFT and other parameter-efficient methods reduce the cost of model parameter updates, they do not address the computational burden of attack generation itself: computing adversarial perturbations still requires iterative forward-backward passes through the full model at each training step.

To address these gaps, we study *adversarial robustness in classification settings* with a focus on token-level suffix attacks, following Howe et al. (2024). We systematically explore strategies for reducing LAT's computation cost from both defense and attack perspectives, both theoretically and empirically. Our contributions are as follows:

- **Defense side:** We introduce LAT-ReFT, a parameter-efficient latent defense that integrates ReFT into LAT. To analyze **(G1)**, as in Figure 1, when the interventions are added on a suffix window of the attacked tokens, the attack success rate is much lower. We further theoretically prove that single-token defense is insufficient: attention mechanisms allow attacks from earlier suffix positions to leak into the defended output (Theorem 1). We then derive conditions under which multi-token suffix window ReFT can provably suppress this leakage (Theorem 2). Empirical results demonstrate the effectiveness of the proposed algorithm in reducing computation costs.
- **Attack side:** Inspired by mechanistic interpretability research showing that model behaviors are determined by sparse circuits (Wang et al., 2022; Elhage et al., 2022; Hanna et al., 2024; Chen et al., 2024), we develop a circuit-guided surrogate attack strategy to overcome **(G2)**: generating attacks on a pruned surrogate model and transferring them to the full model. The key challenge is balancing efficiency and transferability: naive pruning accelerates attack generation but severely degrades attack effectiveness. Inspired by previous work (Molchanov et al., 2016; 2019; Syed et al., 2024), we develop an activation-gradient importance scoring, ActGrad, to identify attack-critical neurons, and provide theoretical support in Theorem 3. Empirical results show that pruning 25% of MLP neurons preserves attack-relevant geometry while reducing the per-step training FLOPs by about 20%. Combined with our defense-side strategy, we achieve an overall computational reduction of 29.6% on average compared to standard LAT with full fine-tuning.
- **Unified insight:** Despite operating through different mechanisms, the defense and attack components reveal a common layerwise pattern: effective ReFT interventions should be placed in early or middle layers rather than very late layers, and the attack surrogate likewise retains most of its important neurons from early and middle layers. This observation is broadly consistent with recent interpretability evidence suggesting that late layers in Transformers often have limited functional complexity (Skean et al., 2025; Queipo-de Llano et al., 2025), and it provides practical guidance for efficient adversarial training design.

2 Related Work

Adversarial Training for LLMs. Adversarial training (AT) is a widely used approach for improving model robustness by optimizing performance under worst-case perturbations (Madry et al., 2017; Shafahi et al., 2019; Wong et al., 2020; Andriushchenko & Flammarion, 2020). In the context of LLMs, AT typically involves generating adversarial prompts in the discrete token space, which incurs high computational cost (Ebrahimi et al., 2018; Zou et al., 2023b; Howe et al., 2024). To address this challenge, recent work has proposed Continuous Adversarial Training (CAT), where adversarial perturbations are applied in the embedding space (Xhonneux et al., 2024; Dékány et al., 2025). This allows efficient gradient-based optimization, reducing the cost of adversarial example generation. Subsequent work further extends this framework to attacking the latent representations in the intermediate LLM layers (Sheshadri et al., 2024; Casper et al., 2024). Recent theoretical analyses also provide justification for the effectiveness of such continuous perturbations (Fu & Wang, 2026).

Representation Engineering. Representation engineering studies how model behavior can be controlled by operating directly on internal activations, rather than by updating all model parameters (Zou et al., 2023a). Prior work has shown that hidden representations in LLMs encode rich semantic and behavioral information, and modifying these representations can effectively steer model outputs (Turner

et al., 2023; Zheng et al., 2024; Lin et al., 2024; Arditi et al., 2024). Within this line of work, representation fine-tuning (ReFT) provides a parameter-efficient adaptation mechanism by learning localized low-rank interventions in activation space (Wu et al., 2024). Subsequent studies further show that such representation-level updates can serve as an effective alternative to full-parameter fine-tuning (Ren et al., 2025; Zeng et al., 2025). This is closely related to our setting, since latent adversarial defense likewise operates on internal representations and thus naturally benefits from such interventions.

Mechanistic Interpretability and Circuits. Mechanistic interpretability aims to identify the internal components and computational pathways responsible for specific model behaviors (Chen et al., 2024; Lee et al., 2024; Sharkey et al., 2025). A recurring finding in this literature is that many behaviors in LLMs are mediated by sparse circuits or localized subnetworks, rather than being uniformly distributed across all parameters (Wang et al., 2022; Hanna et al., 2024; Patel et al., 2025). To study such locality, prior work has developed methods for localizing and validating important components (Vig et al., 2020; Frantar & Alistarh, 2023). In our work, these findings motivate reducing the computation used in adversarial optimization by focusing on the model components that are most relevant to the expression of adversarial behaviors.

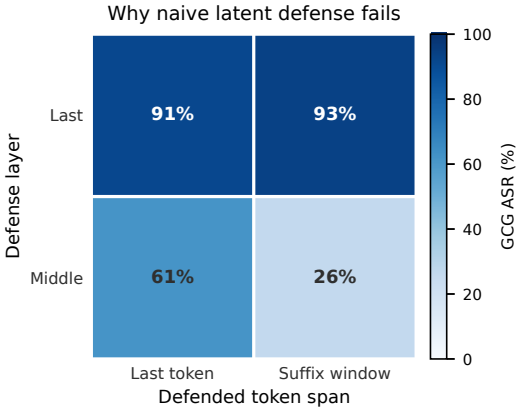

Figure 1: A naive latent defense that protects only the last token at the last layer is ineffective. Robustness improves substantially when the defense covers a suffix window and is moved away from the end of the model.

3 PRELIMINARIES

Transformers. A decoder-only Transformer (Vaswani et al., 2017) maps an input token sequence $x = [x_1, \ldots, x_T]$ to output logits. Let $\mathbf{h}_t^l \in \mathbb{R}^d$ denote the hidden representation of token t at layer $l \in [L_{\text{all}}]$. Each layer contains a self-attention block and an MLP block with residual connections. For notational simplicity, we write the layer update as $\mathbf{h}_t^l = \mathbf{h}_t^{l-1} + \mathbf{a}_t^l + \mathbf{m}_t^l$, where $\mathbf{a}_t^l$ and $\mathbf{m}_t^l$ denote the attention and MLP updates, respectively.

Latent Adversarial Training (LAT). Let $f_\theta(x)$ denote the model logits for input x, parameterized by θ, and let $\ell(f_\theta(x), y)$ be the classification loss for label y. Standard adversarial training seeks parameters that are robust to worst-case perturbations: $\min_\theta \mathbb{E}_{(x,y)} \left[\max_{x' \in \mathcal{B}(x,\varepsilon)} \ell(f_\theta(x'), y)\right]$, where $\mathcal{B}(x, \varepsilon)$ denotes the set of perturbations around x under a perturbation budget ε.

To save the cost of calculating the discrete attacked tokens, LAT (Sheshadri et al., 2024; Casper et al., 2024) instead performs the inner maximization in a continuous hidden space. Let $\mathbf{H}_l(x; \theta) \in \mathbb{R}^{T \times d}$ denote the hidden-state tensor at layer l for the input x. Rather than perturbing the discrete input directly, LAT optimizes over hidden representations in a neighborhood of $\mathbf{H}_l(x; \theta)$:

$$\min_\theta \mathbb{E}_{(x,y)} \left[\ell(f_\theta(x), y) + \lambda_{\text{adv}} \max_{\widetilde{\mathbf{H}} \in \mathcal{B}(\mathbf{H}_l(x;\theta),\varepsilon)} \ell\big(f_\theta(x; \widetilde{\mathbf{H}}), y\big) \right]. \tag{1}$$

Here, $\mathcal{B}(\mathbf{H}_l(x; \theta), \varepsilon)$ denotes the neighborhood in hidden space, and $f_\theta(x; \widetilde{\mathbf{H}})$ denotes the model output when the hidden representation at layer l is replaced by $\widetilde{\mathbf{H}}$. Since the inner optimization is carried out in continuous representation space, it can be efficiently approximated using PGD.

Representation Fine-tuning (ReFT). ReFT (Wu et al., 2024) restricts model updates to a low-rank subspace of the hidden representation. Let $\mathbf{R} \in \mathbb{R}^{r \times d}$ be a matrix with orthonormal rows, where $r \leq d$ is the subspace rank. Given trainable parameters $\mathbf{W} \in \mathbb{R}^{r \times d}$ and $\mathbf{b} \in \mathbb{R}^r$, the ReFT operator applied to a hidden state $\mathbf{h} \in \mathbb{R}^d$ is $\Phi_{\mathbf{R}}(\mathbf{h}) = \mathbf{h} + \mathbf{R}^\top(\mathbf{W}\mathbf{h} + \mathbf{b} - \mathbf{R}\mathbf{h})$.

4 METHOD

Our goal is to make adversarial training for LLMs both effective and efficient. On the defense side, we introduce LAT-ReFT, a parameter-efficient defense that restricts adaptation to a low-rank subspace of the hidden representation. On the attack side, we introduce a circuit-guided surrogate that reduces the cost of inner-loop adversarial optimization while being transferable to the full defended model.

4.1 DEFENSE: LAT-REFT

We integrate ReFT into LAT to obtain a parameter-efficient latent defense by freezing the backbone and training only a low-rank intervention in representation space. Let l_{a} denote the attack layer and l_{r} the defense layer, with $l_{\mathrm{r}} \geq l_{\mathrm{a}}$. Let $\mathcal{T}_{\mathrm{LAT}}$ denote the token positions where latent perturbations are allowed. Consider perturbations $\Delta_{\mathcal{T}} \in \mathbb{R}^{T\times d}$, supported only on $\mathcal{T}_{\mathrm{LAT}}$, i.e., $(\Delta_{\mathcal{T}})_{t,:} = 0$ for all $t \notin \mathcal{T}_{\mathrm{LAT}}$. Thus, the attack is restricted to the selected token positions at layer l_{a}, while ReFT is applied at layer l_{r} to correct the resulting adversarial features. Let $f_{\mathbf{R},\mathbf{W},\mathbf{b}}(x; \widetilde{\mathbf{H}})$ denote the model output when the hidden representation at layer l_{a} is replaced by $\widetilde{\mathbf{H}}$ and the ReFT module parameterized by $(\mathbf{R}, \mathbf{W}, \mathbf{b})$ is applied at layer l_{r}. The defense objective is

$$\min_{\mathbf{R},\mathbf{W},\mathbf{b}} \mathbb{E}_{(x,y)} \left[\ell(f_{\mathbf{R},\mathbf{W},\mathbf{b}}(x), y) + \lambda_{\mathrm{adv}} \max_{\widetilde{\mathbf{H}} \in \mathcal{B}(\mathbf{H}_{l_{\mathrm{a}}}(x;\theta), \varepsilon; \mathcal{T}_{\mathrm{LAT}})} \ell\Big(f_{\mathbf{R},\mathbf{W},\mathbf{b}}(x; \widetilde{\mathbf{H}}), y \Big) \right]. \tag{2}$$

Here, $\mathcal{B}(\mathbf{H}_{l_{\mathrm{a}}}(x;\theta), \varepsilon; \mathcal{T}_{\mathrm{LAT}})$ denotes the hidden-space neighborhood obtained by restricting perturbations to the token positions in $\mathcal{T}_{\mathrm{LAT}}$.

A central design issue for ReFT-based latent defense is *where* to place the intervention. As illustrated in Figure 1, this choice has two parts: (1) selecting the defended suffix window and (2) selecting the intervention layer. We discuss these two design choices in turn below.

Suffix-window defense. A ReFT intervention applied only at the last token is not sufficient in our setting. Empirically, restricting the defense to a single position leaves the model vulnerable to suffix-style attacks, indicating that the intervention is too local. We therefore defend a contiguous suffix window rather than a single token: $\mathcal{T}_{\mathrm{LAT}} = \{T - L + 1, \ldots, T\}$. We provide theoretical support for this design in Section 5. To supplement, we also provide results for prefix/infix attacks and present that a suffix window can still help defend against those attacks; see Appendix C.3.

Defense layer. Prior latent-defense work typically treats layer choice empirically (Sheshadri et al., 2024; Casper et al., 2024), while existing analyses suggest that Transformer layers can be roughly grouped into early, middle, and late stages with different roles (Skean et al., 2025). Guided by both perspectives, we compare intervention layers from these regions in Section 6.3. Our results show that late-layer intervention is generally ineffective, and ReFT should be away from the end of the model.

4.2 CIRCUIT-GUIDED SURROGATE FOR ATTACK GENERATION

Another problem in LAT is the cost of repeatedly solving the inner maximization in Eq. (2). Even though latent optimization is cheaper than discrete token-space search, generating adversarial perturbations still requires many PGD steps with repeated forward–backward passes through a large model. To reduce this cost, we introduce a circuit-guided surrogate.

Specifically, we use a surrogate model for inner-loop attack generation: the latent perturbation is optimized on the surrogate and then applied to the full defended model for training. We construct the surrogate by compressing the MLP blocks of the full model. MLP blocks are a natural target because they contain about two-thirds of a Transformer's parameters and account for much computation. They are also straightforward to compress in a structured way, as their expanded hidden layer is organized neuron-wise, so we can retain only a subset of neurons and prune the rest (Geva et al., 2021). To select neurons to retain, we assign each MLP neuron an importance score inspired by attribution analyses of internal model components, which use activation- and gradient-based signals to estimate component importance (Syed et al., 2024). Concretely, for neuron j in layer l, we define

$$s_{l,j} = \sum_{(x,y)\in\mathcal{D}} \sum_{t} \left| a_{t,j}^{(l)}(x) \, \frac{\partial \ell(f_\theta(x), y)}{\partial a_{t,j}^{(l)}} \right|, \tag{3}$$

where $a_{t,j}^{(l)}(x)$ denotes the activation of neuron j at token t in the expanded hidden layer of the MLP block at layer l, and $\mathcal{D}$ is a small calibration set drawn from the clean training data. As will be discussed in Theorem 3, this score favors neurons that are not only active, but also influential for changing the adversarial loss. We rank neurons globally by this score, retain only the top-scoring fraction, and prune the rest when running the inner-loop attack.

Let f_{surr} denote the surrogate; we run PGD on it to obtain an adversarial hidden-state perturbation

$$\widetilde{\mathbf{H}}^{\star}_{\text{surr}} = \underset{\widetilde{\mathbf{H}} \in \mathcal{B}(\mathbf{H}_{l_{\text{a}}}(x;\theta), \varepsilon; \mathcal{T}_{\text{LAT}})}{\arg\max} \ell\big(f_{\text{surr}}(x; \widetilde{\mathbf{H}}), y\big), \tag{4}$$

and transfer the resulting perturbation back to the full defended model. In Section 6.4, we show that this strategy reduces attack-time computation while transferring well to the full defended model.

5 Theoretical Analysis

In this section, we provide theoretical analysis for the suffix-window defense and the surrogate model. We write $\mathbf{h}_t^{l_{\text{r}}}$ for the hidden vector at token t, i.e., the t-th row of $\mathbf{H}_{l_{\text{r}}}(x;\theta)$. Let $\alpha_{T,t}^l$ and $\hat{\alpha}_{T,t}^l$ denote the attention weights from query position T to token t at layer l in the clean and defended forward passes, respectively, where the clean pass is unperturbed, and the defended pass includes latent perturbation together with ReFT intervention. Here, $\tilde{\mathbf{h}}$ denotes the perturbed hidden state before ReFT, and $\hat{\mathbf{h}}$ denotes the defended hidden state after ReFT. We also define the clean and defended value vectors by $\mathbf{v}_t^l = \mathbf{W}_V^l \mathbf{h}_t^{l-1}$ and $\hat{\mathbf{v}}_t^l = \mathbf{W}_V^l \hat{\mathbf{h}}_t^{l-1}$. Here $\mathbf{W}_Q^l, \mathbf{W}_K^l, \mathbf{W}_V^l$ are the query, key, and value matrices of the attention head at layer l, with $\|\mathbf{W}_Q^l\| \le B_Q,\ \|\mathbf{W}_K^l\| \le B_K,\ \|\mathbf{W}_V^l\| \le B_V$ for some finite constants B_Q, B_K, B_V (Edelman et al., 2022; He & Xing, 2025). Unless otherwise specified, $\|\cdot\|$ denotes the ℓ_2 norm for vectors and the spectral norm for matrices. Additional technical details are deferred to Appendix E.1. Besides, we also impose the following assumption:

Assumption 1 (Local Lipschitz continuity of attention)**.** *In a neighborhood of the clean trajectory, the attention weights at each layer are locally Lipschitz in the query and keys, with constant L_α.*

Assumption 1 ensures that small perturbations in the queries and keys induce controlled changes in the attention weights. It is standard for softmax-based attention in a bounded neighborhood of the clean trajectory (Kim et al., 2021).

5.1 Suffix-Window Defense

We first analyze the limitation of defending only position T. By investigating the layer immediately following the defense layer, namely $l = l_{\text{r}} + 1$, we study how residual perturbations at l_{r} propagate to the attention output $\mathbf{o}_T^l$ at the final position T through attention aggregation. Let $\mathcal{T}_{\text{atk}} = \{T-k+1, \dots, T\}$ denote an attacked suffix index set with $k \ge 1$ such that at layer l_{r} the perturbed hidden states satisfy $\tilde{\mathbf{h}}_t^{l_{\text{r}}} = \mathbf{h}_t^{l_{\text{r}}} + \Delta_t$, for all $t \in \mathcal{T}_{\text{atk}}$, and $\Delta_t = \mathbf{0}$ for $t \notin \mathcal{T}_{\text{atk}}$.

Theorem 1 (Last-token defense is insufficient)**.** *Under Assumption 1, further assume that the ReFT operator $\Phi_{\mathbf{R}}$ is applied only at position T and achieves $\|\hat{\mathbf{h}}_T^{l_{\text{r}}} - \mathbf{h}_T^{l_{\text{r}}}\| \le \varepsilon_T$, where ε_T measures how close the defended hidden state at T is to the clean hidden state after ReFT intervention. Since ReFT is applied only at position T, the remaining attacked suffix positions are left uncorrected: $\hat{\mathbf{h}}_t^{l_{\text{r}}} = \mathbf{h}_t^{l_{\text{r}}} + \Delta_t, t \in \mathcal{T}_{\text{atk}} \setminus \{T\}$. Let $\hat{\mathbf{o}}_T^l$ denote the defended attention output at T in layer l. Let $\mathcal{E}_T := \alpha_{T,T}^l \|\mathbf{W}_V^l\| \varepsilon_T$ and let $\mathcal{E}_{\text{attn}} = \sum_{t=1}^{T} |\hat{\alpha}_{T,t}^l - \alpha_{T,t}^l| \cdot \|\hat{\mathbf{v}}_t^l\|$. Then*

$$\big\|\hat{\mathbf{o}}_T^l - \mathbf{o}_T^l\big\| \ \ge\ \left\| \sum_{t=T-k+1}^{T-1} \alpha_{T,t}^l \, \mathbf{W}_V^l \Delta_t \right\| - \mathcal{E}_{\text{attn}} - \mathcal{E}_T. \tag{5}$$

The first term on the right-hand side of Eq. (5) captures the contribution of upstream perturbations through the value path. The term $\mathcal{E}_T$ reflects imperfect ReFT intervention at position T, while $\mathcal{E}_{\text{attn}}$ captures perturbation-induced changes in the attention weights through the query/key path. Theorem 1 shows that defending only position T does not generally block adversarial information from earlier suffix positions. Beyond this attention-level leakage, a more refined analysis also shows that a ReFT module trained only at T does not transfer to earlier positions (Appendix E.2 and E.5).

In contrast to the above limitation, we next consider a sequence-shared ReFT operator Φ_{shared} trained jointly over the defended suffix window $\mathcal{T}_{\text{LAT}}$.

Theorem 2 (Sequence-shared defense over a suffix window)**.** *Under Assumption 1, suppose Φ_{shared} satisfies $\|\Phi_{\text{shared}}(\tilde{\mathbf{h}}_t^{l_r}) - \mathbf{h}_t^{l_r}\| \le \varepsilon_{\text{LAT}}$, for all $t \in \mathcal{T}_{\text{LAT}}$. Here ε_{LAT} quantifies how accurately the shared ReFT operator restores the clean hidden states across the defended suffix window. Further assume that the defended value vectors satisfy $\|\hat{\mathbf{v}}_t^l\| \le M_V$ for some M_V for any $t \in [T]$. For any token-level attack supported on a suffix set $\mathcal{T}_{\text{atk}}$ such that $\mathcal{T}_{\text{atk}} \subseteq \mathcal{T}_{\text{LAT}}$, the defended attention output at position T in layer l satisfies*

$$\|\hat{\mathbf{o}}_T^l - \mathbf{o}_T^l\| \le \varepsilon_{\text{LAT}}\big(B_V + M_V L_\alpha (B_Q + L B_K)\big). \tag{6}$$

Theorem 2 shows that sequence-shared ReFT over a suffix window controls perturbation propagation through both the value path and the attention-weight path, providing theoretical support for window-based defense rather than single-token defense. The full proof appears in Appendix E.3.

5.2 Effectiveness of the Surrogate Model

We next give an informal theorem explaining why a pruned surrogate can remain effective for inner-loop attack generation. The formal version and proof are presented in Appendix E.4.

Theorem 3 (Surrogate approximation)**.** *Let S denote the set of neurons retained by the surrogate. Under local regularity conditions, with probability at least $1 - 2Kp$ for $p \in (0, 1/(2K))$, the PGD optimization gap between the full defended model and the surrogate over K steps satisfies*

$$\sum_{s=1}^{K} G_s \;\le\; C_{\text{surr}}\,\eta \sum_{s=1}^{K} M_s(S) \;+\; o(K\eta), \tag{7}$$

where G_s is the difference between the one-step improvement in the full inner objective obtained by the full PGD direction and that obtained by the surrogate PGD direction, C_{surr} is a constant depending on the Jacobian bound from the attacked hidden state to the MLP expansion layer, η is the PGD step size, and $w_{s,j} := \left|a_{s,j}^{(l_a)}(x)\, r_{s,j}^{(l_a)}\right|^2$, $M_s(S) := \left(\sum_{j \notin S} w_{s,j}\right)^{1/2}$, where $w_{s,j}$ measures the act×grad contribution of neuron j at step s, $M_s(S)$ is the omitted act×grad mass of the neurons pruned by the surrogate at step s, and $r_{s,j}^{(l_a)}$ denotes the corresponding MLP-path gradient component.

Theorem 3 shows that surrogate quality is controlled by the omitted act×grad mass $M_s(S)$. At a fixed pruning ratio ρ, an effective surrogate should therefore retain neurons with the largest $w_{s,j}$. In contrast, if random pruning discards a fraction ρ of neurons uniformly at random, then for the retained set S_r, $\mathbb{E}\big[M_s(S_r)^2\big] = \rho \sum_j w_{s,j}$. Thus, random pruning removes a constant fraction of the total act×grad mass in expectation, whereas ActGrad pruning preferentially keeps the largest-contributing neurons and therefore yields a smaller omitted mass at the same sparsity level.

6 Experiments

We evaluate the robustness–efficiency tradeoff of LAT-ReFT and test two key design choices: suffix-window placement and ActGrad-based surrogate pruning.

6.1 Experimental Setup

Datasets and Models. Following Howe et al. (2024), we use three classification tasks: IMDB, EnronSpam and PasswordMatch. IMDB and EnronSpam are standard natural-language classification tasks for sentiment analysis and spam detection, respectively, and serve as realistic benchmarks in language understanding settings (Maas et al., 2011; Metsis et al., 2006). PasswordMatch is a procedurally constructed task inspired by TensorTrust (Toyer et al., 2023), and provides a more controlled setting for studying adversarial robustness. We evaluate three pre-trained LLMs: Llama-3.1-8B (Grattafiori et al., 2024), Qwen-2.5-3B (Yang et al., 2024) and Pythia-1.4B (Biderman et al., 2023). For classification tasks, we attach a task-specific linear classification head to the last-token hidden state and fine-tune each model on clean data to obtain a task-adapted classifier for subsequent adversarial training (Howe et al., 2024). Detailed settings are provided in Appendix B.1.

Table 1: Main results on IMDB across three representative models. We report clean accuracy (Acc) and attack success rate (ASR) under RandomToken and GCG. We also report training-efficiency statistics: trainable parameters as a percentage of total model parameters and per-step adversarial-training compute (FLOPs/step). See more results and FLOPs details in Appendix C.1 and F.

		Performance			Efficiency	
Model	**Method**	**Acc↑**	**RandomToken↓**	**GCG↓**	**Params (%)↓**	**FLOPs / step ($\times 10^{12}$)↓**
Llama-3.1-8B	No Defense	0.98	0.93	0.97	–	–
	R2D2	**1.00**	0.08	0.58	100.0	3750.4
	LAT	0.95	**0.00**	**0.02**	100.0	167.0
	CAT	0.98	0.01	0.08	0.5559	184.5
	Ours	0.98	0.05	0.15	**0.0066**	**139.8**
Qwen-2.5-3B	No Defense	**1.00**	0.98	0.99	–	–
	R2D2	0.96	0.07	0.35	100.0	331.2
	LAT	0.95	0.06	**0.10**	100.0	66.4
	CAT	0.97	**0.03**	0.35	0.9608	75.9
	Ours	0.98	0.38	0.26	**0.0085**	**39.2**
Pythia-1.4B	No Defense	0.97	0.73	0.97	–	–
	R2D2	**1.00**	0.05	0.19	100.0	161.7
	LAT	0.96	**0.01**	**0.14**	100.0	28.6
	CAT	0.98	0.04	0.19	0.9505	32.2
	Ours	0.94	0.14	0.33	**0.0203**	**19.6**

Defenses and attack methods. We compare three baselines: R2D2, CAT (Xhonneux et al., 2024), and LAT (Sheshadri et al., 2024; Casper et al., 2024). For R2D2, we follow the version of Howe et al. (2024), which maintains and continually refreshes a pool of previously generated adversarial examples during training rather than generating all attacks from scratch at every step (Mazeika et al., 2024). CAT performs adversarial training in continuous embedding space by applying PGD-based perturbations to input token embeddings, whereas LAT perturbs the model's latent representations in intermediate layers during training. All baselines are implemented in our classification setting; additional details are provided in Appendix B.4. For our method, we implement LAT-ReFT with the ReFT intervention over a defended suffix window and use the ActGrad-pruned surrogate for inner-loop attack generation. The exact details are provided in Appendix B.3. We evaluate robustness primarily under token-level suffix attacks, which match the attack setting assumed by our method and theory. Specifically, we consider attacks that append adversarial suffixes: RandomToken (Howe et al., 2024) and Greedy Coordinate Gradient (GCG) (Zou et al., 2023b) as representatives of random sampling and gradient-based optimization. Additional details are provided in Appendix B.5. The pruned surrogate is used only to generate latent perturbations during training; all evaluation attacks are performed against the full defended model, i.e., the complete backbone together with the trained ReFT intervention.

6.2 Main Results

Table 1 summarizes the main robustness–efficiency results. Full-parameter baselines such as LAT and R2D2 provide strong robustness, but require full-model updates and substantial attack-generation compute; R2D2 may further improve with more training rounds at higher offline cost. CAT reduces the number of trainable parameters while still performing full-model adversarial search. In contrast, our method uses only 0.0066%–0.0203% trainable parameters and reduces per-step FLOPs by 29.6% on average compared with LAT. Implementation-level measurements further confirm these savings, with 12.7–31.8% lower step time and 56.6–73.8% lower peak GPU memory than LAT (Appendix F.2). As expected, this efficiency gain comes with a loss in absolute robustness compared with full-parameter LAT. For example, on Llama-3.1-8B, LAT achieves the lowest GCG ASR of 0.02, while our method obtains 0.15 with lower FLOPs and only 0.0066% trainable parameters. RandomToken is more model-dependent. On Qwen-2.5-3B, our method reduces ASR from 0.98 to 0.38, but remains less effective than under GCG. One possible reason is that RandomToken repeatedly samples perturbations over the suffix, inducing a broader and less gradient-aligned attack distribution than the perturbations used during training. These random suffixes appear to exploit a negative-to-positive class bias not fully exposed by GCG. Overall, LAT-ReFT with a circuit-guided surrogate provides a lightweight

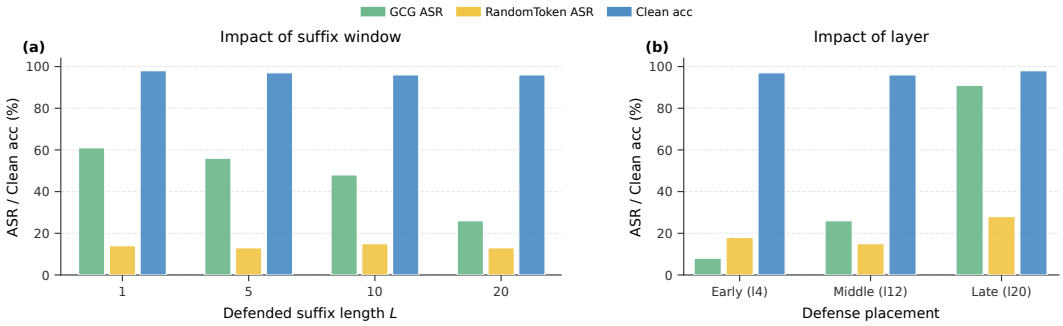

Figure 2: Defense-placement ablations on IMDB with Pythia-1.4B under GCG and RandomToken attacks. (a) Suffix-window length ablation with the defense layer fixed at $l = 12$. (b) Defense-layer ablation with the suffix length fixed at $L = 20$. Defending a suffix window and avoiding very late intervention layers both improve robustness, especially under GCG.

alternative to full-parameter adversarial training with a favorable robustness–efficiency tradeoff, which is particularly useful when compute budgets make full-parameter adversarial training impractical.

6.3 Design Analysis of the Defense

We study the two factors of LAT-ReFT mentioned in Section 4.1: the suffix window and the defense layer. Figure 2 summarizes the corresponding ablations.

Suffix-window size. Figure 2(a) shows that defending only the final suffix token is insufficient, which is consistent with Theorem 1. Specifically, as the defended suffix length L increases from 1 to 20, the green bars (GCG ASR) decrease substantially, from 61% to 26%. This indicates that broader suffix coverage is important for suppressing adversarial leakage from nearby attacked positions, which aligns with Theorem 2. By contrast, yellow bars (RandomToken ASR) remain low and vary little with L, likely because its random perturbations are less consistently aligned with the suffix leakage path characterized in Theorem 1. The blue bars (clean accuracy) also remain unchanged, showing that the robustness gains under GCG are not driven by sacrificing clean performance.

Defense layer. Figure 2(b) shows that placing the defense too late in the model is ineffective. Specifically, the green bars (GCG ASR) are much higher for late-layer intervention than for early or middle placement, while the yellow bars (RandomToken ASR) show the same qualitative trend at a lower overall level. In contrast, the blue bars (clean accuracy) remain consistently high across all three placements. To explain the worse performance of late layers, the adversarial perturbation has already propagated through many layers by the time it reaches the final layer, leaving less room for a low-rank intervention to remove it reliably. This is broadly consistent with recent work showing that decoder-only Transformers organize computation into distinct depth-wise phases, with later layers acting more as selective refinement stages (Queipo-de Llano et al., 2025).

Overall, these results support both parts of our placement strategy: defending a suffix window rather than a single token, and avoiding very late-layer intervention when applying the latent defense.

6.4 Analysis of Circuit-Guided Surrogate

We next analyze whether the circuit-guided surrogate can reduce inner-loop attack cost while preserving attack transferability. Figure 3 evaluates this design from three aspects: (1) the pruning-ratio tradeoff between training cost and attack success rate, (2) the impact of different neuron-selection rules, and (3) the resulting layer-wise pruning patterns.

Tradeoff between pruning and ASR. We first study the tradeoff between the pruning ratio and ASR on IMDB with Pythia-1.4B under GCG attack as a representative setting. Figure 3(a) reports the resulting ASR together with the total FLOPs per adversarial training step. The results show that moderate pruning remains effective: at 25% pruning, the surrogate reduces FLOPs to $0.804\times$

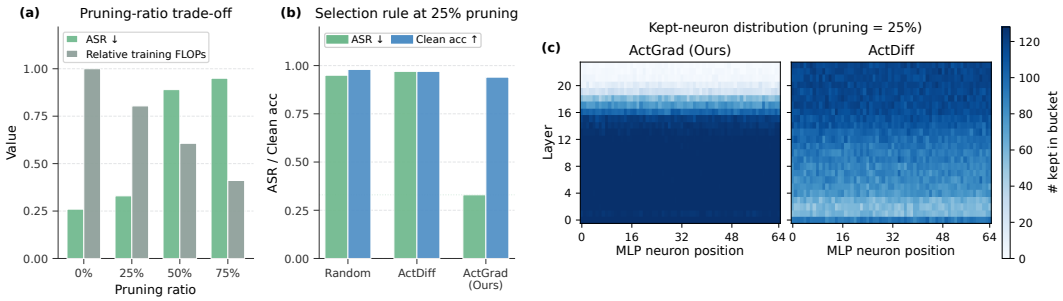

Figure 3: Analysis of the circuit-guided surrogate on IMDB with Pythia-1.4B under GCG attack. (a) Tradeoff between ASR and normalized training FLOPs across pruning ratios. (b) Selection-rule comparison at 25% pruning. ActGrad strongly outperforms random pruning and activation-difference scoring in ASR. (c) Layer-by-neuron patterns at 25% pruning for ActGrad and ActDiff.

that of the unpruned model, while ASR increases only slightly from 0.26 to 0.33. In contrast, more aggressive pruning sharply degrades transfer: at 50% pruning, ASR rises to 0.89, and at 75% pruning it further increases to 0.95. We therefore use a pruning ratio of 25% in all remaining experiments. This degradation is expected: the surrogate is useful only if its PGD directions transfer to the full defended model. Moderate pruning preserves these adversarial search directions, whereas aggressive pruning distorts them and weakens transfer.

Selection rule comparison. We next compare three neuron-selection rules at the same pruning ratio of 25%: random selection (an unstructured pruning baseline), activation-difference scoring (ActDiff), and our activation-times-gradient (ActGrad) rule. This tests whether surrogate quality depends on which neurons are retained, rather than only on the pruning ratio. ActDiff is motivated by the difference-in-means style of representation analysis widely used in mechanistic interpretability and steering-vector methods (Marks & Tegmark, 2023). Figure 3(b) shows that the distinction is crucial. While the clean accuracies remain high for all methods, at the same pruning ratio, random pruning gives ASR 0.95, and ActDiff performs similarly poorly at 0.97. In contrast, ActGrad reduces ASR to 0.33. This supports the effectiveness of ActGrad and aligns with Theorem 3.

Neuron pruning patterns. Figure 3(c) provides a more detailed view of the resulting patterns at the default 25% pruning ratio, using layer-by-neuron heatmaps for ActGrad and ActDiff. The two methods produce different structures: ActGrad preserves more capacity in earlier and middle layers, with a relatively sharp transition toward heavier pruning only in later layers. By contrast, ActDiff allocates its budget more diffusely and places comparatively more mass in later-layer regions. Together with the results in Figure 3(b), this suggests that, in our setting, preserving earlier- and middle-layer MLP computation is more important for maintaining transferable adversarial directions. This may explain why ActDiff performs slightly worse than random pruning: its retained computation is less concentrated in earlier and middle layers. See more details in Appendix D.

7 Conclusion

We proposed an efficient latent adversarial training framework for LLMs that improves the robustness–efficiency tradeoff from both defense and attack sides. On the defense side, LAT-ReFT combines LAT with ReFT and shows that effective intervention should cover a suffix window and avoid very late layers. On the attack side, our circuit-guided surrogate retains high-importance MLP neurons using activation-gradient scores. Together, these components provide a lightweight alternative to full-parameter adversarial training, supported by theoretical analysis and empirical evidence. While our study focuses on LLM-based classifiers, the computational savings of our framework are not tied to the classification head, and extending its robustness evaluation to open-ended generation is a natural next step.

AI USE STATEMENT

We used generative AI tools (LLM-based assistants) for language editing of the manuscript, including grammar correction and improving the clarity of phrasing. We also used such tools to assist with writing and debugging parts of the experimental code. We did not use generative AI tools to generate research ideas, design the proposed methods, or produce or analyze the experimental results. All AI-assisted text was reviewed and revised by the authors, and all AI-assisted code was manually inspected and tested by the authors before use. All references were manually verified by the authors. The language models evaluated in this paper (Llama-3.1-8B, Qwen-2.5-3B and Pythia-1.4B) are objects of study. We take full responsibility for the final content of this work.

REFERENCES

Akshit Achara and Anshuman Chhabra. Watching the ai watchdogs: A fairness and robustness analysis of ai safety moderation classifiers. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pp. 253–264, 2025.

Maksym Andriushchenko and Nicolas Flammarion. Understanding and improving fast adversarial training. *Advances in Neural Information Processing Systems*, 33:16048–16059, 2020.

Maksym Andriushchenko, Francesco Croce, and Nicolas Flammarion. Jailbreaking leading safety-aligned llms with simple adaptive attacks. *arXiv preprint arXiv:2404.02151*, 2024.

Andy Arditi, Oscar Obeso, Aaquib Syed, Daniel Paleka, Nina Panickssery, Wes Gurnee, and Neel Nanda. Refusal in language models is mediated by a single direction. *Advances in Neural Information Processing Systems*, 37:136037–136083, 2024.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pp. 2397–2430. PMLR, 2023.

Stephen Casper, Lennart Schulze, Oam Patel, and Dylan Hadfield-Menell. Defending against unforeseen failure modes with latent adversarial training. *arXiv preprint arXiv:2403.05030*, 2024.

Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George J Pappas, and Eric Wong. Jailbreaking black box large language models in twenty queries. In *2025 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, pp. 23–42. IEEE, 2025.

Jianhui Chen, Xiaozhi Wang, Zijun Yao, Yushi Bai, Lei Hou, and Juanzi Li. Towards understanding safety alignment: A mechanistic perspective from safety neurons. *arXiv preprint arXiv:2406.14144*, 2024.

Csaba Dékány, Stefan Balauca, Robin Staab, Dimitar I Dimitrov, and Martin Vechev. Mixat: Combining continuous and discrete adversarial training for llms. *arXiv preprint arXiv:2505.16947*, 2025.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. Hotflip: White-box adversarial examples for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 31–36, 2018.

Benjamin L Edelman, Surbhi Goel, Sham Kakade, and Cyril Zhang. Inductive biases and variable creation in self-attention mechanisms. In *International Conference on Machine Learning*, pp. 5793–5831. PMLR, 2022.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, et al. Toy models of superposition. *arXiv preprint arXiv:2209.10652*, 2022.

Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International conference on machine learning*, pp. 10323–10337. PMLR, 2023.

Shaopeng Fu and Di Wang. Understanding and improving continuous llm adversarial training via in-context learning theory. *International Conference on Learning Representations*, 2026.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 5484–5495, 2021.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

William Hackett, Lewis Birch, Stefan Trawicki, Neeraj Suri, and Peter Garraghan. Bypassing llm guardrails: An empirical analysis of evasion attacks against prompt injection and jailbreak detection systems. In *Proceedings of the The First Workshop on LLM Security (LLMSEC)*, pp. 101–114, 2025.

Michael Hanna, Sandro Pezzelle, and Yonatan Belinkov. Have faith in faithfulness: Going beyond circuit overlap when finding model mechanisms. *arXiv preprint arXiv:2403.17806*, 2024.

Robert Hart. The ai security nightmare is here and it looks suspiciously like lobster. https://www.theverge.com/ai-artificial-intelligence/881574/cline-openclaw-prompt-injection-hack, February 2026. The Verge, accessed April 19, 2026.

Weiyi He and Yue Xing. Impact of positional encoding: Clean and adversarial rademacher complexity for transformers under in-context regression. *arXiv preprint arXiv:2512.09275*, 2025.

Nikolaus Howe, Ian McKenzie, Oskar Hollinsworth, Michał Zajac, Tom Tseng, Aaron Tucker, Pierre-Luc Bacon, and Adam Gleave. Scaling trends in language model robustness. *arXiv preprint arXiv:2407.18213*, 2024.

Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, et al. Llama guard: Llm-based input-output safeguard for human-ai conversations. *arXiv preprint arXiv:2312.06674*, 2023.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Hyunjik Kim, George Papamakarios, and Andriy Mnih. The lipschitz constant of self-attention. In *International Conference on Machine Learning*, pp. 5562–5571. PMLR, 2021.

Andrew Lee, Xiaoyan Bai, Itamar Pres, Martin Wattenberg, Jonathan K Kummerfeld, and Rada Mihalcea. A mechanistic understanding of alignment algorithms: A case study on dpo and toxicity. *arXiv preprint arXiv:2401.01967*, 2024.

Yuping Lin, Pengfei He, Han Xu, Yue Xing, Makoto Yamada, Hui Liu, and Jiliang Tang. Towards understanding jailbreak attacks in llms: A representation space analysis. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 7067–7085, 2024.

Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pp. 142–150, 2011.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Samuel Marks and Max Tegmark. The geometry of truth: Emergent linear structure in large language model representations of true/false datasets. *arXiv preprint arXiv:2310.06824*, 2023.

Mantas Mazeika, Long Phan, Xuwang Yin, Andy Zou, Zifan Wang, Norman Mu, Elham Sakhaee, Nathaniel Li, Steven Basart, Bo Li, et al. Harmbench: A standardized evaluation framework for automated red teaming and robust refusal. *arXiv preprint arXiv:2402.04249*, 2024.

Vangelis Metsis, Ion Androutsopoulos, and Georgios Paliouras. Spam filtering with naive bayes-which naive bayes? In *CEAS*, volume 17, pp. 28–69. Mountain View, CA, 2006.

Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. *arXiv preprint arXiv:1611.06440*, 2016.

Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11264–11272, 2019.

Pravin Nair. Softmax is $1/2$-lipschitz: A tight bound across all ℓ_p norms. *arXiv preprint arXiv:2510.23012*, 2025.

Dev Patel, Gabrielle Gervacio, Diekola Raimi, Kevin Zhu, Ryan Lagasse, Gabriel Grand, Ashwinee Panda, and Maheep Chaudhary. Alignment-constrained dynamic pruning for llms: Identifying and preserving alignment-critical circuits. *arXiv preprint arXiv:2511.07482*, 2025.

Enrique Queipo-de Llano, Álvaro Arroyo, Federico Barbero, Xiaowen Dong, Michael Bronstein, Yann LeCun, and Ravid Shwartz-Ziv. Attention sinks and compression valleys in llms are two sides of the same coin. *arXiv preprint arXiv:2510.06477*, 2025.

Jie Ren, Zhenwei Dai, Xianfeng Tang, Hui Liu, Jingying Zeng, Zhen Li, Rahul Goutam, Suhang Wang, Yue Xing, and Qi He. A general framework to enhance fine-tuning-based llm unlearning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 18464–18476, 2025.

Ali Shafahi, Mahyar Najibi, Mohammad Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free! *Advances in neural information processing systems*, 32, 2019.

Lee Sharkey, Bilal Chughtai, Joshua Batson, Jack Lindsey, Jeff Wu, Lucius Bushnaq, Nicholas Goldowsky-Dill, Stefan Heimersheim, Alejandro Ortega, Joseph Bloom, et al. Open problems in mechanistic interpretability. *arXiv preprint arXiv:2501.16496*, 2025.

Abhay Sheshadri, Aidan Ewart, Phillip Guo, Aengus Lynch, Cindy Wu, Vivek Hebbar, Henry Sleight, Asa Cooper Stickland, Ethan Perez, Dylan Hadfield-Menell, et al. Latent adversarial training improves robustness to persistent harmful behaviors in llms. *arXiv preprint arXiv:2407.15549*, 2024.

Oscar Skean, Md Rifat Arefin, Dan Zhao, Niket Patel, Jalal Naghiyev, Yann LeCun, and Ravid Shwartz-Ziv. Layer by layer: Uncovering hidden representations in language models. *arXiv preprint arXiv:2502.02013*, 2025.

Aaquib Syed, Can Rager, and Arthur Conmy. Attribution patching outperforms automated circuit discovery. In *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pp. 407–416, 2024.

Sam Toyer, Olivia Watkins, Ethan Adrian Mendes, Justin Svegliato, Luke Bailey, Tiffany Wang, Isaac Ong, Karim Elmaaroufi, Pieter Abbeel, Trevor Darrell, et al. Tensor trust: Interpretable prompt injection attacks from an online game. *arXiv preprint arXiv:2311.01011*, 2023.

Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J Vazquez, Ulisse Mini, and Monte MacDiarmid. Steering language models with activation engineering. *arXiv preprint arXiv:2308.10248*, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. Investigating gender bias in language models using causal mediation analysis. *Advances in neural information processing systems*, 33:12388–12401, 2020.

Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small. *arXiv preprint arXiv:2211.00593*, 2022.

Eric Wong, Leslie Rice, and J Zico Kolter. Fast is better than free: Revisiting adversarial training. *arXiv preprint arXiv:2001.03994*, 2020.

Zhengxuan Wu, Aryaman Arora, Zheng Wang, Atticus Geiger, Dan Jurafsky, Christopher D Manning, and Christopher Potts. Reft: Representation finetuning for language models. *Advances in Neural Information Processing Systems*, 37:63908–63962, 2024.

Sophie Xhonneux, Alessandro Sordoni, Stephan Günnemann, Gauthier Gidel, and Leo Schwinn. Efficient adversarial training in llms with continuous attacks. *Advances in Neural Information Processing Systems*, 37:1502–1530, 2024.

Cihang Xie and Alan Yuille. Intriguing properties of adversarial training at scale. *arXiv preprint arXiv:1906.03787*, 2019.

Yue Xing, Qifan Song, and Guang Cheng. On the algorithmic stability of adversarial training. *Advances in neural information processing systems*, 34:26523–26535, 2021.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Jiahao Yu, Xingwei Lin, Zheng Yu, and Xinyu Xing. Gptfuzzer: Red teaming large language models with auto-generated jailbreak prompts. *arXiv preprint arXiv:2309.10253*, 2023.

Shenglai Zeng, Pengfei He, Kai Guo, Tianqi Zheng, Hanqing Lu, Yue Xing, and Hui Liu. Towards context-robust llms: A gated representation fine-tuning approach. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10262–10276, 2025.

Mengnan Zhao, Lihe Zhang, Jingwen Ye, Huchuan Lu, Baocai Yin, and Xinchao Wang. Adversarial training: A survey. *arXiv preprint arXiv:2410.15042*, 2024.

Chujie Zheng, Fan Yin, Hao Zhou, Fandong Meng, Jie Zhou, Kai-Wei Chang, Minlie Huang, and Nanyun Peng. Prompt-driven llm safeguarding via directed representation optimization. *arXiv preprint arXiv:2401.18018*, 3, 2024.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, et al. Representation engineering: A top-down approach to ai transparency. *arXiv preprint arXiv:2310.01405*, 2023a.

Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*, 2023b.

APPENDIX

A Limitations and Future Work **15**

B Additional Experimental Setup **15**

B.1 Datasets and Fine-tuning 15

B.2 Evaluation Metric 16

B.3 LAT-ReFT Training Setup 16

B.4 Baseline Defense Configurations 16

B.5 Adversarial Attacks Setup 18

B.6 Computation Cost 18

C Additional Results **18**

C.1 More Results 18

C.2 Additional Surrogate Analysis 19

C.3 Prefix and Infix Attacks 19

C.4 Effect of ReFT Rank 20

D Circuit-Guided Surrogate Details **20**

D.1 ActDiff Method 20

D.2 Surrogate pruning heatmaps 20

E Additional Theory and Proofs **23**

E.1 Complete Mathematical Setup 23

E.2 Proof of Theorem 1 23

E.3 Proof of Theorem 2 25

E.4 Proof of Theorem 3 25

E.5 A Spatial Transfer Limitation of Last-Token ReFT 28

F Estimated Compute Calculations **28**

F.1 Estimated FLOPs 28

F.2 Measured Compute Time and Peak Memory 30

A Limitations and Future Work

In this work, we focus on the adversarial robustness of LLM-based classifiers, following Howe et al. (2024). This setting offers an unambiguous notion of attack success and is of direct practical relevance, since classifiers such as moderation and content filters are routinely deployed as safeguards in security-critical settings (Inan et al., 2023; Achara & Chhabra, 2025). Studying jailbreaks on open-ended tasks, however, requires generative models. Because the inner-loop attack and the outer-loop update that our method accelerates do not depend on the task head, we expect its computational savings to carry over to generative settings, whereas whether the robustness gains transfer requires a dedicated evaluation across generative models and jailbreak attacks, which we leave to future work.

Next, our evaluation follows the suffix-based threat model of Howe et al. (2024), with preliminary results under prefix and infix attacks in Appendix C.3. Although our GCG evaluation is white-box and differentiates through the ReFT intervention, it would be valuable to evaluate stronger adaptive attacks that explicitly exploit knowledge of the defense, such as the location of the defended token window or the intervention layer, as well as attacks with larger compute budgets.

Finally, several design choices in our method could be made adaptive. We defend a fixed suffix window and use a fixed MLP pruning ratio for the surrogate; selecting the defended token positions adaptively, or using adaptive pruning and broader circuit extraction, may further improve the robustness–efficiency tradeoff.

B Additional Experimental Setup

B.1 Datasets and Fine-tuning

We use 3 binary classification datasets (Howe et al., 2024). The two natural-language datasets are filtered to examples with 100 to 1,000 GPT-2 tokens. The short-text datasets are filtered to examples with 5 to 50 tokens. From the length-filtered training split of each dataset, we sample up to 20,000 examples with a fixed random seed ($s = 42$) to form the fine-tuning training set (`ft_train`). From the remaining training examples not selected into `ft_train`, we further sample up to 100 examples as the attack set, which is disjoint from `ft_train` by construction. For evaluation, we apply the same length filter to the validation split (100–1,000 tokens for natural-language datasets; 5–50 tokens for short-text datasets) and retain *all* examples that pass the filter without further subsampling. Table 2 summarizes the resulting split sizes.

Table 2: Dataset statistics after length filtering.

Dataset	Train (raw)	Train (filtered)	`ft_train`	Attack	Val (raw)	Val (filtered)
IMDB	24,365	22,446	20,000	100	24,401	22,327
EnronSpam	29,290	20,733	20,000	100	1,852	1,289
PasswordMatch	25,000	25,000	20,000	100	25,000	25,000

We fine-tune each model by attaching a two-class linear classification head on top of the final hidden state of the last token and training end-to-end. All models are trained for 3 epochs using AdamW with a learning rate of 10^{-5} and linear decay to zero (no warmup). The effective batch size is 16 for all models (per-device batch size 4 with 4 gradient accumulation steps for Llama-3.1-8B). Input sequences are truncated to 512 tokens for the natural-language datasets (IMDB, Enron) and to 64 tokens for the short-text datasets (PasswordMatch). 8B-scale models are trained in bfloat16. Table 3 reports validation accuracy of the resulting classifiers.

Table 3: Fine-tuned classifier validation accuracy (%).

Dataset	**Pythia-1.4B**	**Qwen2.5-3B**	**Llama3.1-8B**
IMDB	94.4	96.5	97.1
Enron	97.8	98.4	99.2
PasswordMatch	100.0	100.0	100.0

B.2 Evaluation Metric

We evaluate each defended classifier using clean accuracy and attack success rate (ASR). Clean accuracy is computed on the unperturbed test set. For adversarial evaluation, we report ASR over the attack set: the fraction of these examples that are correctly classified before attack and misclassified after attack.

B.3 LAT-ReFT Training Setup

Algorithm 1 summarizes one training iteration of LAT-ReFT with the circuit-guided surrogate. For the main IMDB experiments, we train LoReFT interventions on top of task-specific fine-tuned classifiers initialized from the corresponding clean baseline checkpoint. Table 4 lists the hyperparameters used for the reported IMDB results with ReFT rank=64 for all models.

Table 4: LAT-ReFT hyperparameters used for reported IMDB results.

Model	ReFT layer	Attack layer	Pos.	ϵ	PGD steps	λ_{adv}
Pythia-1.4B	12	4	l20	0.2	8	1.0
Llama-3.1-8B	8	4	l20	0.5	8	1.0
Qwen2.5-3B	18	18	l20	0.5	8	1.0

Table 5: LAT-ReFT hyperparameters for the cross-dataset results on Pythia-1.4B.

Dataset	ReFT layer	Attack layer	Pos.	ϵ	λ_{adv}
EnronSpam	12	12	l20	0.2	1.0
PasswordMatch	18	18	l20	0.3	1.0

B.4 Baseline Defense Configurations

R2D2. We follow the R2D2 adversarial training setup of Howe et al. (2024) (Algorithm 1), including its dynamic adversarial pool construction and mixed clean/adversarial minibatch sampling strategy. Each experiment runs for R adversarial training rounds, where R is chosen per model based on the available compute budget; specifically $R=5$ for Llama-3.1-8B, $R=4$ for Pythia-1.4B, and $R=7$ for Qwen-2.5-3B. In each round we attack n_{adv}=200 examples drawn from the attack split using GCG with a suffix of N=10 tokens, B=128 candidates per iteration, beam width k=256, and a linearly increasing iteration budget $k(r) = \text{round}\Big(k_{\text{start}} + \frac{r}{R_{\max}}(k_{\text{end}} - k_{\text{start}})\Big)$ with k_{start}=8, k_{end}=32, $R_{\max}$=8, yielding $k \in \{11, 14, 17, 20, 23, 26, 29, 32\}$ across rounds. Newly found adversarial examples are added to a persistent pool and resampled via exponential rank-weighting (λ=0.005) that jointly prioritises high-loss and recently generated examples. Each round's fine-tuning minibatch contains n_{aug}=1000 items (80% adversarial, 20% clean), trained for 200 gradient steps with AdamW (lr $= 2\times10^{-5}$, batch size 8, no weight decay; for Llama-3.1-8B we use lr $= 5\times10^{-6}$ and batch size 4 to prevent clean-accuracy collapse). Total FLOPs reported in Table 1 are computed as $C_{\text{adv}} = C_{\text{search}} + C_{\text{train}}$ using the Kaplan formula $C = 6ND$ with model parameter counts $N \in \{1.31\text{B}, 3.09\text{B}, 7.50\text{B}\}$ for Pythia-1.4B, Qwen-2.5-3B, and Llama-3.1-8B respectively; training FLOPs account for less than 3% of the total in all cases.

CAT. Continuous Adversarial Training (CAT) applies projected gradient descent (PGD) *in the embedding space* at every training step. Given a mini-batch with token embeddings $\mathbf{E} \in \mathbb{R}^{B\times T\times d}$, the inner maximization runs k PGD steps within an ℓ_2 ball of radius ε:

$$\boldsymbol{\delta}^* = \underset{\|\boldsymbol{\delta}\|_2 \le \varepsilon}{\arg\max}\ \mathcal{L}(f_\theta(\mathbf{E} + \boldsymbol{\delta}),\, y)\,,$$

and the outer minimization updates the model via

$$\min_\theta\ \mathcal{L}_{\text{clean}} + \lambda\, \mathcal{L}_{\text{adv}}.$$

Algorithm 1 One iteration of LAT-ReFT with circuit-guided surrogate

Require: Pretrained model backbone f_θ, surrogate f_{surr}, attack layer l_{a}, defense layer l_{r}, defended token set $\mathcal{T}_{\text{LAT}}$, PGD steps K, perturbation budget ε, adversarial weight λ_{adv}, ReFT parameters $(\mathbf{R}, \mathbf{W}, \mathbf{b})$, learning rate η

Freeze the backbone parameters θ; optimize only the ReFT parameters $(\mathbf{R}, \mathbf{W}, \mathbf{b})$ at layer l_{r}
for each minibatch (x, y) **do**
  Compute clean hidden states $\mathbf{H}_{l_{\text{a}}}(x; \theta)$
  Compute clean loss

$$\mathcal{L}_{\text{clean}} \leftarrow \ell(f_{\mathbf{R},\mathbf{W},\mathbf{b}}(x), y)$$

  Construct the surrogate with the current ReFT parameters at layer l_{r}
  Initialize $\widetilde{\mathbf{H}}^{(0)} \in \mathcal{B}(\mathbf{H}_{l_{\text{a}}}(x; \theta), \varepsilon; \mathcal{T}_{\text{LAT}})$
  for $s = 0, \ldots, K-1$ **do**
    Compute surrogate inner-loss gradient

$$g^{(s)} \leftarrow \nabla_{\widetilde{\mathbf{H}}} \ell\Big(f_{\text{surr}}(x; \widetilde{\mathbf{H}}^{(s)}), y\Big)$$

    Take one PGD step and project back to the latent attack set

$$\widetilde{\mathbf{H}}^{(s+1)} \leftarrow \Pi_{\mathcal{B}(\mathbf{H}_{l_{\text{a}}}(x;\theta),\varepsilon;\mathcal{T}_{\text{LAT}})}\Big(\widetilde{\mathbf{H}}^{(s)} + \alpha\, \text{sign}(g^{(s)})\Big)$$

  end for
  Set $\widetilde{\mathbf{H}}^{\star} \leftarrow \widetilde{\mathbf{H}}^{(K)}$
  Compute adversarial loss on the full defended model

$$\mathcal{L}_{\text{adv}} \leftarrow \ell\Big(f_{\mathbf{R},\mathbf{W},\mathbf{b}}(x; \widetilde{\mathbf{H}}^{\star}), y\Big)$$

  Form the outer objective

$$\mathcal{L}_{\text{total}} \leftarrow \mathcal{L}_{\text{clean}} + \lambda_{\text{adv}}\, \mathcal{L}_{\text{adv}}$$

  Update only the ReFT parameters

$$(\mathbf{R}, \mathbf{W}, \mathbf{b}) \leftarrow (\mathbf{R}, \mathbf{W}, \mathbf{b}) - \eta\, \nabla_{(\mathbf{R},\mathbf{W},\mathbf{b})} \mathcal{L}_{\text{total}}$$

end for

In our implemented CAT baseline, we use LoRA adapters on top of a 4-bit quantized backbone rather than full-model fine-tuning, following Xhonneux et al. (2024). For the final runs, we use LoRA rank $r = 16$, LoRA scaling $\alpha = 32$, $\varepsilon = 0.05$, PGD step size 0.005, $k = 10$, adversarial weight $\lambda = 0.5$, learning rate 10^{-4}, and batch size 8.

LAT. Latent Adversarial Training (LAT) performs adversarial training by injecting perturbations into an intermediate hidden representation rather than the input token embeddings. For an input-label pair (x, y), let $\mathbf{H}_{l_{\text{a}}}(x)$ denote the hidden states at transformer layer l_{a}. The inner maximization searches for a latent perturbation within an ℓ_2 ball:

$$\boldsymbol{\delta}^* = \arg\max_{\|\boldsymbol{\delta}\|_2 \le \varepsilon} \mathcal{L}\Big(f_\theta^{(l_{\text{a}})}(\mathbf{H}_{l_{\text{a}}}(x) + \boldsymbol{\delta}),\, y\Big),$$

where $f_\theta^{(l_{\text{a}})}$ denotes the remainder of the model after injecting the perturbed activation at layer l_{a}. We solve this inner problem with projected gradient descent (PGD) using random initialization:

$$\boldsymbol{\delta} \leftarrow \Pi_{\|\boldsymbol{\delta}\|_2 \le \varepsilon}\left(\boldsymbol{\delta} + \alpha \frac{\nabla_{\boldsymbol{\delta}} \mathcal{L}_{\text{adv}}}{\|\nabla_{\boldsymbol{\delta}} \mathcal{L}_{\text{adv}}\|_2}\right).$$

The outer objective is

$$\min_\theta\ \mathcal{L}_{\text{clean}} + \lambda\, \mathcal{L}_{\text{adv}}.$$

In our LAT baseline, we fine-tune all model parameters and restrict the latent perturbation to a fixed subset of positions: the final 20 non-padding tokens at the selected transformer layer, matching the

suffix-focused attack setting used in our discrete attacks. For the final IMDB runs, we use $k = 8$ PGD steps, $\alpha = \varepsilon/5$, and $\lambda = 1.0$. The learning rate is 2×10^{-5} for Pythia-1.4B and Qwen-2.5-3B, and 1×10^{-5} for Llama-3.1-8B. We report the best layer–radius setting from the LAT sweep: layer 4 with $\varepsilon = 1.0$ for Llama-3.1-8B, layer 8 with $\varepsilon = 1.5$ for Qwen-2.5-3B, and layer 4 with $\varepsilon = 0.5$ for Pythia-1.4B.

B.5 Adversarial Attacks Setup

We evaluate robustness using two adversarial attacks under different suffix-based search strategies. All attacks are evaluated on 100 samples drawn from the designated attack split of the dataset. We report clean accuracy (ACC) and Attack Success Rate (ASR).

GCG (Greedy Coordinate Gradient). We use token-level `GCG` (Zou et al., 2023b) in suffix mode: N=10 adversarial tokens are appended to the original input. Each round computes the gradient of the cross-entropy loss with respect to the input embeddings, selects the top-k=256 candidate replacement tokens per attack position, randomly samples 128 of these candidates for forward-pass evaluation, and applies the single token substitution that maximises the loss. The attack runs for T=10 rounds.

RandomToken. `RandomToken` is a gradient-free baseline designed to be comparable to `GCG` in the number of adversarial tokens while replacing gradient-guided search with uniform random sampling, following Algorithm 2 of Howe et al. (2024). At each iteration, all N=10 adversarial suffix tokens are replaced simultaneously by tokens drawn uniformly at random from the non-special vocabulary. The attack evaluates the resulting prompt with one forward pass and stops immediately upon a successful flip; otherwise it retains the best-loss candidate seen across all iterations. Given that an iteration of `RandomToken` is much cheaper than an iteration of `GCG`, we use $T = 500$ iterations.

Threat model scope. Our main evaluation focuses on token-level suffix attacks, which are directly aligned with both the adversarial training setting studied in this paper and the perturbation model used in our theoretical analysis. We do not include semantic rewriting attacks such as PAIR (Chao et al., 2025) in the main evaluation, since they operate under a broader threat model: they use an external attacker model to iteratively rewrite the input itself. In our classification setting, this would allow the attacker to freely reformulate the original example, making the attack no longer comparable to suffix-based perturbations.

B.6 Computation Cost

Experiments were run on GPU clusters using NVIDIA H200, L40S, and RTX A6000 GPUs. Most jobs used a single GPU worker; CPU workers were used for data loading, preprocessing, and job orchestration. The exact GPU type varied across runs due to shared-cluster scheduling. For the main experiments, we report method-level compute using FLOPs in Table 1. Overall, we estimate that the full project required on the order of several hundred GPU-hours, approximately 800 GPU-hours in total. Measured wall-clock time and peak memory are reported in Appendix F.2.

C Additional Results

C.1 More Results

PasswordMatch inputs are short (12–20 tokens), so an N=10 token suffix is a much larger relative perturbation than on the natural-language datasets; we therefore use a defended window (L=20) that exceeds the suffix length, consistent with the trend in Figure 2(a). The remaining difficulty is not specific to our method: full-parameter LAT reaches a GCG ASR of 0.88, compared with 0.52 for ours, suggesting that latent-space adversarial training in general transfers poorly to discrete suffixes comparable in length to the prompt. R2D2, which trains directly on GCG-generated adversarial text, remains the strongest defense on this task.

Table 6: Additional cross-dataset robustness results on Pythia-1.4B. We report clean accuracy (Acc, higher is better) and attack success rate (ASR, lower is better) under RandomToken and GCG suffix attacks.

Dataset	Method	Acc↑	RandomToken ASR↓	GCG ASR↓
EnronSpam	No Defense	0.99	0.55	0.96
	R2D2	**1.00**	0.03	0.11
	LAT	**1.00**	0.11	0.29
	Ours	0.99	**0.02**	**0.08**
PasswordMatch	No Defense	0.92	0.92	0.92
	R2D2	**1.00**	**0.00**	**0.04**
	LAT	**1.00**	0.51	0.88
	Ours	**1.00**	0.52	0.52

Table 7: Clean accuracy and GCG attack success rate at different pruning ratios of the surrogate MLP neurons on Pythia-1.4B. The surrogate is constructed by pruning neurons ranked by the proposed activation-gradient importance score. 0% corresponds to the full unpruned model. Lower GCG ASR is better; higher clean accuracy is better.

Dataset	Pruning ratio	Clean Acc↑	GCG ASR↓
IMDB	0% (full)	0.96	**0.26**
	25%	0.94	0.33
	50%	**0.97**	0.89
EnronSpam	0% (full)	**0.99**	0.26
	25%	0.99	**0.08**
	50%	0.99	0.76

C.2 Additional Surrogate Analysis

Table 7 shows that the optimal pruning ratio can vary across datasets. Moderate pruning sometimes improves transferability over the full surrogate, as in EnronSpam, suggesting that pruning may remove task-irrelevant or noisy neurons and produce a cleaner adversarial search direction. This is consistent with the circuit-sparsity motivation behind our surrogate construction. At the same time, aggressive pruning substantially weakens transfer, indicating that too much pruning destroys attack-relevant geometry.

C.3 Prefix and Infix Attacks

The suffix-window defense is motivated by the classifier readout structure rather than by suffix attacks alone. In decoder-only sequence classification, the final prediction is primarily read from the last hidden state. We therefore expect defending a suffix window near this readout position to remain useful even against prefix attacks.

To verify this, we evaluate GCG on IMDB with Pythia-1.4B, placing the same number of adversarial tokens (N=10) at the prefix, at an infix position located 90% of the way through the input sequence (Howe et al., 2024), or at the suffix, with all other attack settings as in Appendix B.5. As shown in Table 8, LAT-ReFT reduces the ASR from 0.97 to 0.26 under suffix attacks and from 0.58 to 0.34 under infix attacks. For prefix attacks, the undefended model already has a relatively low ASR of 0.26, and our method maintains a comparable ASR of 0.24.

This pattern is consistent with the readout structure: since the prediction is read from the final hidden state, perturbations at earlier positions reach the readout only through self-attention and are therefore less effective, which yields the prefix $<$ infix $<$ suffix ordering of ASR in the undefended model. For the same reason, a suffix-window intervention near the readout also mitigates part of the downstream effect of prefix and infix perturbations.

Table 8: GCG ASR under different adversarial token positions on IMDB with Pythia-1.4B.

Method	Suffix ASR↓	Prefix ASR↓	Infix ASR↓
No Defense	0.97	0.26	0.58
LAT-ReFT	**0.26**	**0.24**	**0.34**

C.4 Effect of ReFT Rank

We examine the effect of the ReFT rank on IMDB with Llama-3.1-8B under GCG, keeping all other settings fixed (Table 9). Very small ranks provide insufficient capacity to correct adversarially shifted representations and lead to high ASR, whereas the ASR decreases substantially as the rank grows. This is consistent with prior evidence that adversarial training benefits more from increased model capacity than standard training (Xie & Yuille, 2019), while low ranks are often sufficient for standard downstream adaptation (Wu et al., 2024). We therefore adopt rank 64 as a unified default across models and datasets; further increasing the rank increases the number of trainable parameters and computation, yielding a capacity–efficiency trade-off.

Table 9: Effect of ReFT rank on IMDB with Llama-3.1-8B under GCG.

Rank	Params (%)↓	Clean Acc↑	GCG ASR↓
4	0.0004	0.98	0.94
16	0.0017	0.97	0.87
32	0.0033	0.98	0.56
64	0.0066	0.98	**0.15**

D Circuit-Guided Surrogate Details

D.1 ActDiff Method

ActDiff is motivated by the difference-in-means style of representation analysis widely used in mechanistic interpretability and steering-vector methods. Concretely, it ranks neurons by the absolute class-conditional gap in mean activation magnitude:

$$s_{l,j}^{\mathrm{ActDiff}} = \left| \frac{1}{|\mathcal{D}_+|} \sum_{x \in \mathcal{D}_+} \sum_t \left| a_{t,j}^{(l)}(x) \right| - \frac{1}{|\mathcal{D}_-|} \sum_{x \in \mathcal{D}_-} \sum_t \left| a_{t,j}^{(l)}(x) \right| \right|, \tag{8}$$

where $\mathcal{D}_+, \mathcal{D}_-$ are the positive and negative calibration subsets. Unlike ActGrad, however, this rule depends only on clean activation statistics and does not measure how much a neuron actually affects the attack objective.

D.2 Surrogate pruning heatmaps

We further visualize the layerwise pruning patterns for two representative datasets. These heatmaps illustrate how the pruned MLP neurons are distributed across layers for different model families.

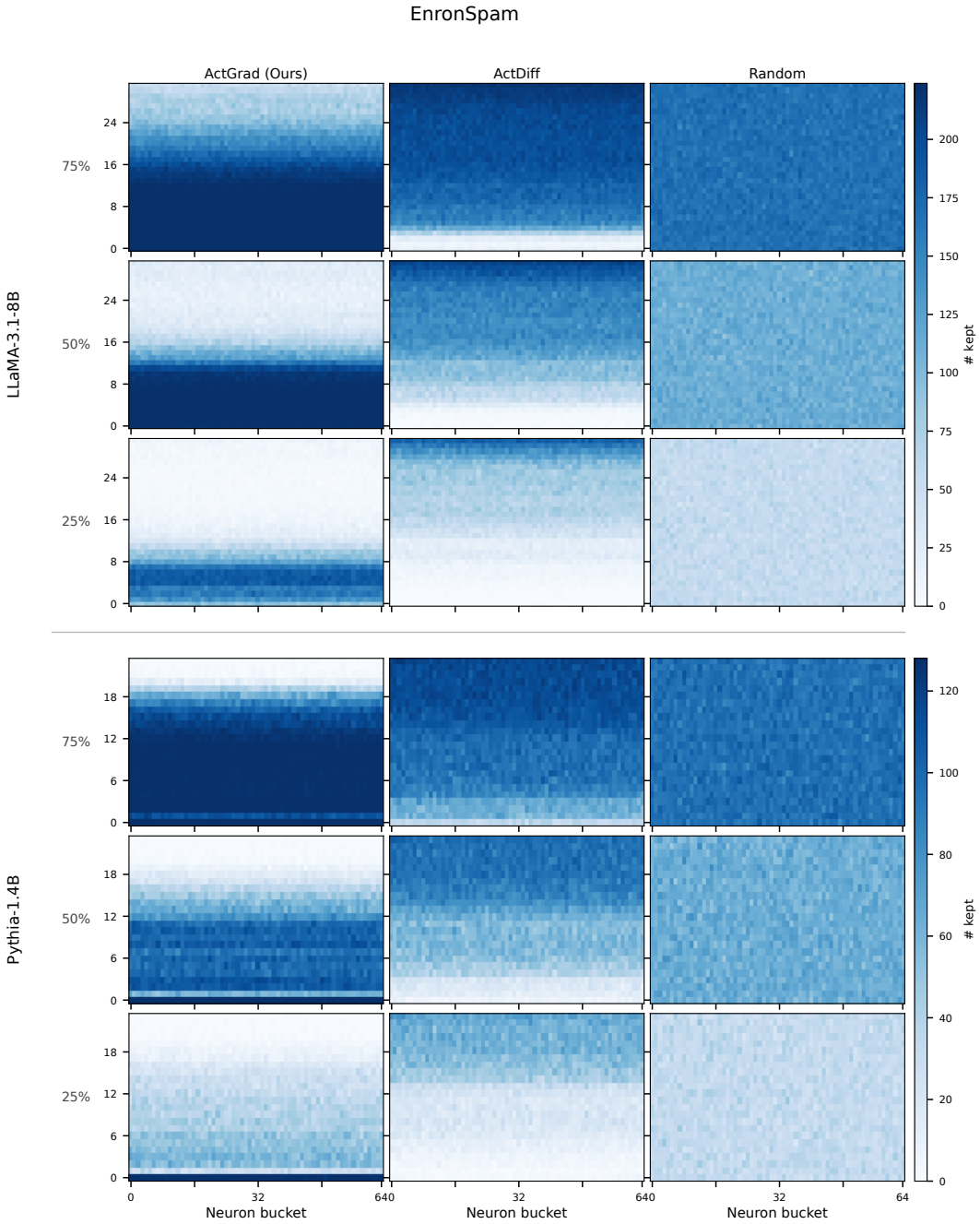

Figure 4: Heatmaps of layerwise surrogate pruning patterns on EnronSpam. 75% denotes retaining 75% of MLP neurons (i.e., pruning 25%).

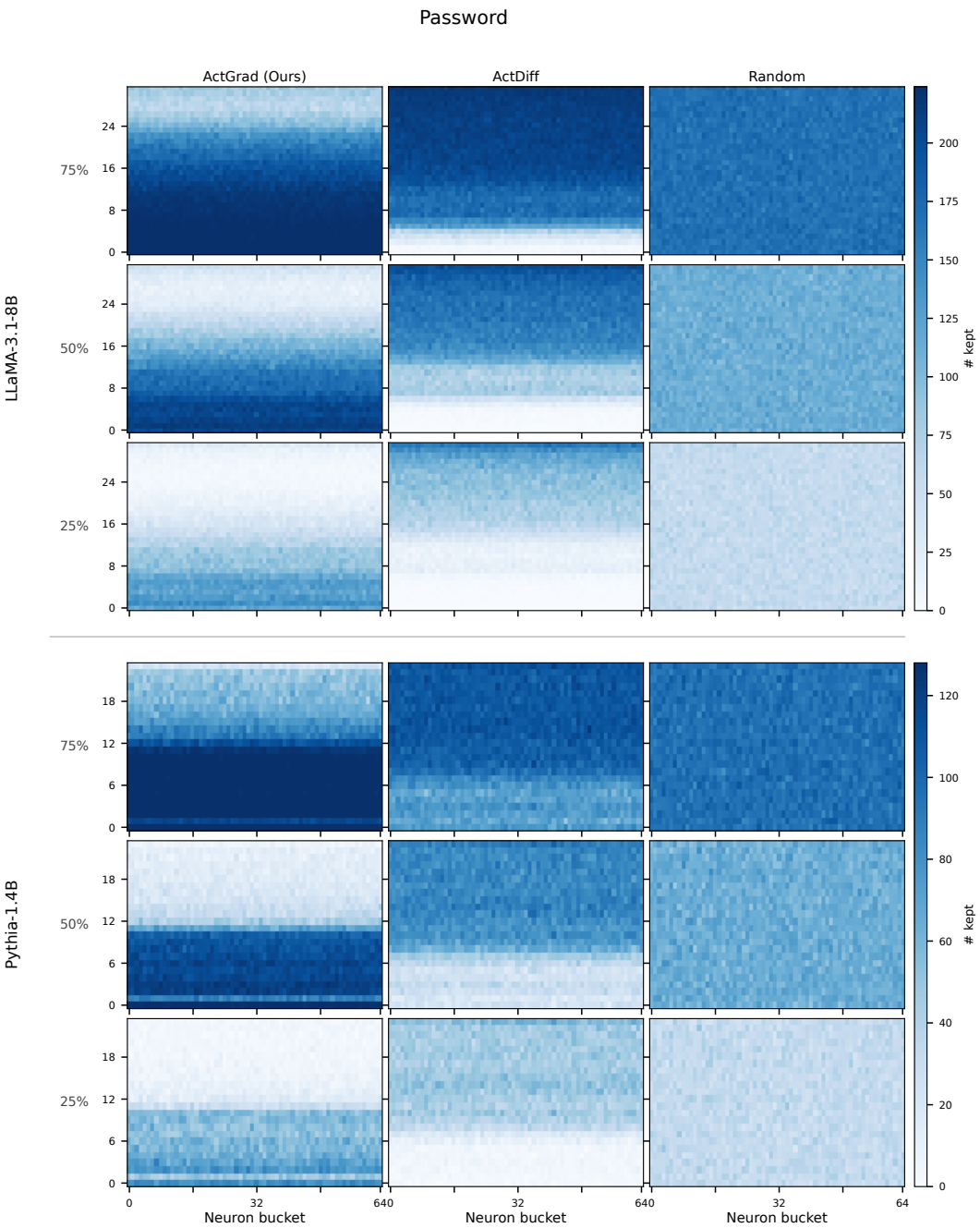

Figure 5: Heatmaps of layerwise surrogate pruning patterns on PasswordMatch.

E Additional Theory and Proofs

E.1 Complete Mathematical Setup

We provide the full attention-level setup used in the theoretical analysis. As in the main text, we analyze a single attention head at a fixed layer $l = l_{\mathrm{r}} + 1$, since the multi-head case only changes constants. For the final token at position T, define

$$\mathbf{q}_T^l = \mathbf{W}_Q^l \mathbf{h}_T^{l_{\mathrm{r}}}, \quad \mathbf{k}_t^l = \mathbf{W}_K^l \mathbf{h}_t^{l_{\mathrm{r}}}, \quad \mathbf{v}_t^l = \mathbf{W}_V^l \mathbf{h}_t^{l_{\mathrm{r}}},$$

and let the corresponding attention output be

$$\mathbf{o}_T^l = \sum_{t=1}^{T} \alpha_{T,t}^l \mathbf{v}_t^l, \quad \alpha_{T,t}^l = \frac{\exp\Big((\mathbf{q}_T^l)^\top \mathbf{k}_t^l / \sqrt{d}\Big)}{\sum_{j=1}^{T} \exp\Big((\mathbf{q}_T^l)^\top \mathbf{k}_j^l / \sqrt{d}\Big)}.$$

For completeness, we also restate the ReFT operator used throughout the proofs. Let $\mathbf{R} \in \mathbb{R}^{r \times d}$ be row-orthonormal, i.e.,

$$\mathbf{R}\mathbf{R}^\top = \mathbf{I}_r,$$

and let $\mathbf{W} \in \mathbb{R}^{r \times d}$ and $\mathbf{b} \in \mathbb{R}^r$. For a hidden state $\mathbf{h} \in \mathbb{R}^d$, the ReFT intervention is

$$\Phi_{\mathbf{R}}(\mathbf{h}) = \mathbf{h} + \mathbf{R}^\top(\mathbf{W}\mathbf{h} + \mathbf{b} - \mathbf{R}\mathbf{h}) = (\mathbf{I} - \mathbf{R}^\top\mathbf{R})\mathbf{h} + \mathbf{R}^\top(\mathbf{W}\mathbf{h} + \mathbf{b}).$$

Since $\mathbf{R}$ has orthonormal rows, $\mathbf{R}^\top\mathbf{R}$ is the orthogonal projector onto the edited subspace $\mathrm{span}(\mathbf{R})$, and

$$\Pi_{\mathrm{span}(\mathbf{R})^\perp} := \mathbf{I} - \mathbf{R}^\top\mathbf{R}$$

is the orthogonal projector onto its orthogonal complement.

E.2 Proof of Theorem 1

Proof. We analyze the attention layer $l = l_{\mathrm{r}} + 1$ immediately after the defense layer. Let $(\mathbf{q}_T^l, \mathbf{k}_t^l, \mathbf{v}_t^l, \mathbf{o}_T^l)$ and $(\hat{\mathbf{q}}_T^l, \hat{\mathbf{k}}_t^l, \hat{\mathbf{v}}_t^l, \hat{\mathbf{o}}_T^l)$ denote the clean and defended query, key, value, and attention output, respectively.

For attacked suffix positions $t \in \mathcal{T}_{\mathrm{atk}} \setminus \{T\}$, no correction is applied, so

$$\hat{\mathbf{h}}_t^{l_{\mathrm{r}}} = \mathbf{h}_t^{l_{\mathrm{r}}} + \Delta_t,$$

which implies

$$\hat{\mathbf{k}}_t^l = \mathbf{k}_t^l + \mathbf{W}_K^l \Delta_t, \quad \hat{\mathbf{v}}_t^l = \mathbf{v}_t^l + \mathbf{W}_V^l \Delta_t.$$

At the defended token T, the ReFT correction gives

$$\big\|\hat{\mathbf{h}}_T^{l_{\mathrm{r}}} - \mathbf{h}_T^{l_{\mathrm{r}}}\big\| \le \varepsilon_T,$$

and therefore

$$\|\hat{\mathbf{q}}_T^l - \mathbf{q}_T^l\| \le \|\mathbf{W}_Q^l\|\varepsilon_T, \quad \|\hat{\mathbf{k}}_T^l - \mathbf{k}_T^l\| \le \|\mathbf{W}_K^l\|\varepsilon_T, \quad \|\hat{\mathbf{v}}_T^l - \mathbf{v}_T^l\| \le \|\mathbf{W}_V^l\|\varepsilon_T.$$

We decompose the output difference as

$$\begin{aligned}
\hat{\mathbf{o}}_T^l - \mathbf{o}_T^l &= \sum_{t=1}^{T} \alpha_{T,t}^l (\hat{\mathbf{v}}_t^l - \mathbf{v}_t^l) + \sum_{t=1}^{T} (\hat{\alpha}_{T,t}^l - \alpha_{T,t}^l)\hat{\mathbf{v}}_t^l \\
&= \sum_{t=T-k+1}^{T-1} \alpha_{T,t}^l \mathbf{W}_V^l \Delta_t + \alpha_{T,T}^l (\hat{\mathbf{v}}_T^l - \mathbf{v}_T^l) + \sum_{t=1}^{T} (\hat{\alpha}_{T,t}^l - \alpha_{T,t}^l)\hat{\mathbf{v}}_t^l.
\end{aligned}$$

Applying the reverse triangle inequality gives

$$\begin{aligned}
\big\|\hat{\mathbf{o}}_T^l - \mathbf{o}_T^l\big\| &\ge \left\| \sum_{t=T-k+1}^{T-1} \alpha_{T,t}^l \mathbf{W}_V^l \Delta_t \right\| - \alpha_{T,T}^l \|\hat{\mathbf{v}}_T^l - \mathbf{v}_T^l\| - \left\| \sum_{t=1}^{T} (\hat{\alpha}_{T,t}^l - \alpha_{T,t}^l)\hat{\mathbf{v}}_t^l \right\| \\
&\ge \left\| \sum_{t=T-k+1}^{T-1} \alpha_{T,t}^l \mathbf{W}_V^l \Delta_t \right\| - \alpha_{T,T}^l \|\mathbf{W}_V^l\|\, \varepsilon_T - \sum_{t=1}^{T} \big|\hat{\alpha}_{T,t}^l - \alpha_{T,t}^l\big| \cdot \|\hat{\mathbf{v}}_t^l\|.
\end{aligned}$$

This proves Eq. (5) with

$$\mathcal{E}_T = \alpha^l_{T,T}\|\mathbf{W}^l_V\|\,\varepsilon_T, \quad \mathcal{E}_{\text{attn}} = \sum_{t=1}^{T}\big|\hat{\alpha}^l_{T,t} - \alpha^l_{T,t}\big| \cdot \|\hat{\mathbf{v}}^l_t\|.$$

It remains to bound $\mathcal{E}_{\text{attn}}$. The attention weights depend on the query $\mathbf{q}^l_T$ and the key collection $\{\mathbf{k}^l_t\}_{t=1}^T$ through the softmax map. Under Assumption 1, there exists a local Lipschitz constant $L_\alpha > 0$ such that

$$\sum_{t=1}^{T}\big|\hat{\alpha}^l_{T,t} - \alpha^l_{T,t}\big| \le L_\alpha\left(\|\hat{\mathbf{q}}^l_T - \mathbf{q}^l_T\| + \sum_{t=1}^{T}\|\hat{\mathbf{k}}^l_t - \mathbf{k}^l_t\|\right).$$

Using the bounds established above,

$$\|\hat{\mathbf{q}}^l_T - \mathbf{q}^l_T\| \le \|\mathbf{W}^l_Q\|\,\varepsilon_T,$$

and

$$\sum_{t=1}^{T}\|\hat{\mathbf{k}}^l_t - \mathbf{k}^l_t\| \le \|\mathbf{W}^l_K\|\,\varepsilon_T + \|\mathbf{W}^l_K\|\sum_{t=T-k+1}^{T-1}\|\Delta_t\|.$$

Therefore,

$$\sum_{t=1}^{T}\big|\hat{\alpha}^l_{T,t} - \alpha^l_{T,t}\big| \le L_\alpha\left(\|\mathbf{W}^l_Q\|\,\varepsilon_T + \|\mathbf{W}^l_K\|\,\varepsilon_T + \|\mathbf{W}^l_K\|\sum_{t=T-k+1}^{T-1}\|\Delta_t\|\right).$$

If the defended trajectory remains in a bounded neighborhood of the clean one, then there exists a constant $M_V > 0$ such that

$$\sup_{1\le t\le T}\|\hat{\mathbf{v}}^l_t\| \le M_V.$$

Hence

$$\begin{aligned}\mathcal{E}_{\text{attn}} &\le M_V\sum_{t=1}^{T}\big|\hat{\alpha}^l_{T,t} - \alpha^l_{T,t}\big| \\ &\le M_V L_\alpha\left((\|\mathbf{W}^l_Q\| + \|\mathbf{W}^l_K\|)\varepsilon_T + \|\mathbf{W}^l_K\|\sum_{t=T-k+1}^{T-1}\|\Delta_t\|\right).\end{aligned}$$

This bound on $\mathcal{E}_{\text{attn}}$ is used in Corollary 1 below to identify when the lower bound in Theorem 1 is non-trivial. □

Corollary 1 (Non-trivial leakage under last-token defense)**.** *In the setting of Theorem 1, further assume $\|\hat{\mathbf{v}}^l_t\| \le M_V$ for all $t \in [T]$, and that the uncorrected upstream perturbations share a common direction, i.e., $\Delta_t = \delta_t\mathbf{u}$ with $\|\mathbf{u}\| = 1$ and $\delta_t \ge 0$ for $t \in \mathcal{T}_{\text{atk}} \setminus \{T\}$. Define*

$$\gamma := \|\mathbf{W}^l_V\mathbf{u}\|\min_{t\in\mathcal{T}_{\text{atk}}\setminus\{T\}}\alpha^l_{T,t}, \qquad \kappa := M_V L_\alpha\|\mathbf{W}^l_K\|.$$

Then

$$\big\|\hat{\mathbf{o}}^l_T - \mathbf{o}^l_T\big\| \;\ge\; (\gamma - \kappa)\sum_{t=T-k+1}^{T-1}\delta_t \;-\; \Big(\alpha^l_{T,T}\|\mathbf{W}^l_V\| + M_V L_\alpha\big(\|\mathbf{W}^l_Q\| + \|\mathbf{W}^l_K\|\big)\Big)\varepsilon_T.$$

In particular, if $\gamma > \kappa$, the deviation of the defended attention output grows linearly with the magnitude of the upstream perturbations, even when the last token is restored exactly ($\varepsilon_T = 0$).

Proof. Since $\alpha^l_{T,t} \ge 0$ and $\delta_t \ge 0$,

$$\Big\|\sum_{t=T-k+1}^{T-1}\alpha^l_{T,t}\mathbf{W}^l_V\Delta_t\Big\| = \|\mathbf{W}^l_V\mathbf{u}\|\sum_{t=T-k+1}^{T-1}\alpha^l_{T,t}\delta_t \ge \gamma\sum_{t=T-k+1}^{T-1}\delta_t.$$

Combining this with $\mathcal{E}_T = \alpha^l_{T,T}\|\mathbf{W}^l_V\|\varepsilon_T$ and the bound on $\mathcal{E}_{\text{attn}}$ derived above (with $\|\Delta_t\| = \delta_t$) in Eq. (5) yields the claim.

□

E.3 Proof of Theorem 2

Proof. Because $k \le L$, the attack window is contained in the defended suffix window, i.e.,

$$\mathcal{T}_{\mathrm{atk}} \subseteq \mathcal{T}_{\mathrm{LAT}}.$$

Hence, for every attacked position $t \in \mathcal{T}_{\mathrm{atk}}$, the sequence-shared operator satisfies

$$\|\Phi_{\mathrm{shared}}(\tilde{\mathbf{h}}_t^{l_{\mathrm{r}}}) - \mathbf{h}_t^{l_{\mathrm{r}}}\| \le \varepsilon_{\mathrm{LAT}}.$$

Therefore, the corresponding query, key, and value perturbations are bounded by

$$\|\hat{\mathbf{q}}_T^l - \mathbf{q}_T^l\| \le B_Q \varepsilon_{\mathrm{LAT}}, \quad \|\hat{\mathbf{k}}_t^l - \mathbf{k}_t^l\| \le B_K \varepsilon_{\mathrm{LAT}}, \quad \|\hat{\mathbf{v}}_t^l - \mathbf{v}_t^l\| \le B_V \varepsilon_{\mathrm{LAT}}.$$

We decompose the attention distortion into a value term and an attention-weight term:

$$\|\hat{\mathbf{o}}_T^l - \mathbf{o}_T^l\| \le \sum_{t=1}^{T} \alpha_{T,t}^l \|\hat{\mathbf{v}}_t^l - \mathbf{v}_t^l\| + \sum_{t=1}^{T} |\hat{\alpha}_{T,t}^l - \alpha_{T,t}^l| \, \|\hat{\mathbf{v}}_t^l\|.$$

For the value term, only positions in $\mathcal{T}_{\mathrm{LAT}}$ contribute, and $\sum_t \alpha_{T,t}^l = 1$, so

$$\sum_{t=1}^{T} \alpha_{T,t}^l \|\hat{\mathbf{v}}_t^l - \mathbf{v}_t^l\| \le B_V \varepsilon_{\mathrm{LAT}}.$$

For the attention-weight term, Assumption 1 gives

$$\sum_{t=1}^{T} |\hat{\alpha}_{T,t}^l - \alpha_{T,t}^l| \le L_\alpha \left(\|\hat{\mathbf{q}}_T^l - \mathbf{q}_T^l\| + \sum_{t=1}^{T} \|\hat{\mathbf{k}}_t^l - \mathbf{k}_t^l\| \right).$$

Positions outside $\mathcal{T}_{\mathrm{LAT}}$ are neither perturbed nor intervened on, so $\hat{\mathbf{k}}_t^l = \mathbf{k}_t^l$ there. For $t \in \mathcal{T}_{\mathrm{LAT}}$, the ReFT operator is applied, and the assumption on Φ_{shared} gives $\|\hat{\mathbf{h}}_t^{l_{\mathrm{r}}} - \mathbf{h}_t^{l_{\mathrm{r}}}\| \le \varepsilon_{\mathrm{LAT}}$ (for unattacked $t \in \mathcal{T}_{\mathrm{LAT}}$, $\tilde{\mathbf{h}}_t^{l_{\mathrm{r}}} = \mathbf{h}_t^{l_{\mathrm{r}}}$). Hence $\|\hat{\mathbf{k}}_t^l - \mathbf{k}_t^l\| \le B_K \varepsilon_{\mathrm{LAT}}$, and since $|\mathcal{T}_{\mathrm{LAT}}| = L$,

$$\sum_{t=1}^{T} \|\hat{\mathbf{k}}_t^l - \mathbf{k}_t^l\| \le L B_K \varepsilon_{\mathrm{LAT}}.$$

Therefore,

$$\sum_{t=1}^{T} |\hat{\alpha}_{T,t}^l - \alpha_{T,t}^l| \le L_\alpha \varepsilon_{\mathrm{LAT}} (B_Q + L B_K).$$

Assuming $\|\hat{\mathbf{v}}_t^l\| \le M_V$, we obtain

$$\sum_{t=1}^{T} |\hat{\alpha}_{T,t}^l - \alpha_{T,t}^l| \, \|\hat{\mathbf{v}}_t^l\| \le M_V L_\alpha \varepsilon_{\mathrm{LAT}} (B_Q + L B_K).$$

Combining the two bounds yields

$$\|\hat{\mathbf{o}}_T^l - \mathbf{o}_T^l\| \le \varepsilon_{\mathrm{LAT}} \big(B_V + M_V L_\alpha (B_Q + L B_K) \big),$$

which proves the theorem. □

E.4 Proof of Theorem 3

Lemma 1. *To simplify notation, write the MLP expansion layer at l_{a} as $Z = \phi(H^{\mathrm{attn}} W_1)$, where*

$$H^{\mathrm{attn}} = \sigma\left(\mathrm{RowSoftmax}\left(\frac{\widetilde{\mathbf{H}} W_{QK} \widetilde{\mathbf{H}}^\top}{\sqrt{d_m}} \right) \widetilde{\mathbf{H}} W_V \right), \quad W_{QK} = W_Q W_K^\top,$$

and σ and ϕ are L_σ- and L_ϕ-Lipschitz, respectively. Assume that

$$\|W_Q\| \le B_Q, \quad \|W_K\| \le B_K, \quad \|W_V\| \le B_V.$$

Then there exists $C_{\mathrm{attn}} = O(T)$ such that

$$\left\| \frac{\partial Z}{\partial \widetilde{\mathbf{H}}} \right\| \le L_\phi L_\sigma \|W_1\| \left(C_{\mathrm{attn}} \frac{B_Q B_K B_V}{\sqrt{d_m}} + B_V \right).$$

Proof of Lemma 1. By the chain rule,

$$\frac{\partial Z}{\partial \widetilde{\mathbf{H}}} = \frac{\partial Z}{\partial H^{\text{attn}}} \frac{\partial H^{\text{attn}}}{\partial \widetilde{\mathbf{H}}}.$$

Since ϕ is L_ϕ-Lipschitz,

$$\left\| \frac{\partial Z}{\partial H^{\text{attn}}} \right\| \le L_\phi \|W_1\|.$$

Let $A(\widetilde{\mathbf{H}}) = \text{RowSoftmax}\Big(\frac{\widetilde{\mathbf{H}} W_{QK} \widetilde{\mathbf{H}}^\top}{\sqrt{d_m}}\Big)$, so that $H^{\text{attn}} = \sigma\Big(A(\widetilde{\mathbf{H}})\, \widetilde{\mathbf{H}} W_V\Big)$. Since σ is L_σ-Lipschitz,

$$\left\| \frac{\partial H^{\text{attn}}}{\partial \widetilde{\mathbf{H}}} \right\| \le L_\sigma \left\| \frac{\partial \big(A(\widetilde{\mathbf{H}}) \widetilde{\mathbf{H}} W_V\big)}{\partial \widetilde{\mathbf{H}}} \right\|.$$

For a single-row softmax $p = \text{softmax}(z)$, its Jacobian is

$$D\,\text{softmax}(z) = \text{diag}(p) - pp^\top,$$

and satisfies the uniform spectral-norm bound $\|D\,\text{softmax}(z)\| \le \frac{1}{2}$ (Nair, 2025). Therefore the RowSoftmax map is uniformly Lipschitz row-wise. Then the derivative of $A(\widetilde{\mathbf{H}})\, \widetilde{\mathbf{H}} W_V$ with respect to $\widetilde{\mathbf{H}}$ consists of a value-path term and an attention-weight term. The value-path term is bounded by B_V. For the attention-weight term, the softmax Jacobian bound together with

$$\|W_{QK}\| \le \|W_Q\|\,\|W_K\| \le B_Q B_K, \quad \|W_V\| \le B_V,$$

gives a contribution proportional to $\frac{\|W_{QK}\|\|W_V\|}{\sqrt{d_m}}$, multiplied by a sequence-length accumulation factor. Absorbing this factor into $C_{\text{attn}} = O(T)$, we obtain

$$\left\| \frac{\partial H^{\text{attn}}}{\partial \widetilde{\mathbf{H}}} \right\| \le L_\sigma \left(C_{\text{attn}} \frac{B_Q B_K B_V}{\sqrt{d_m}} + B_V \right).$$

Combining the above displays gives

$$\left\| \frac{\partial Z}{\partial \widetilde{\mathbf{H}}} \right\| \le L_\phi L_\sigma \|W_1\| \left(C_{\text{attn}} \frac{B_Q B_K B_V}{\sqrt{d_m}} + B_V \right).$$

□

Assumption 2 (Non-degeneracy on pruned active coordinates)**.** *There exist $m_0 > 0$ and $p \in (0,1)$ such that, with probability at least $1-p$,*

$$\min_{j \notin S} \left| a_{t,j}^{(l_a)}(x) \right| \ge m_0,$$

for all $t \in [K]$.

Assumption 3 (Local sign-stability)**.** *There exists a local constant $c > 0$ such that, on the same event,*

$$\langle g_t, u_t \rangle - \langle g_t, u_t^s \rangle \le c\, \|g_t - g_t^s\|,$$

for all $t \in [K]$, where

$$g_t = \nabla_{\widetilde{\mathbf{H}}} L_f\Big(\widetilde{\mathbf{H}}^{(t)}\Big), \quad g_t^s = \nabla_{\widetilde{\mathbf{H}}} L_s\Big(\widetilde{\mathbf{H}}^{(t)}\Big),$$

and $u_t := \text{sign}(g_t), u_t^s := \text{sign}(g_t^s)$ be the one-step ℓ_∞-PGD ascent directions.

Proof of Theorem 3. We work on the event on which Assumptions 2 and 3 hold. Define

$$G_t := L_f\Big(\widetilde{\mathbf{H}}^{(t)} + \eta u_t\Big) - L_f\Big(\widetilde{\mathbf{H}}^{(t)} + \eta u_t^s\Big).$$

Let Z denote the MLP expansion layer at l_{a}, and let $r_t := \nabla_Z L_f\left(\widetilde{\mathbf{H}}^{(t)}\right)$. Write P_S for the projection onto the retained neuron set S. Define

$$s_{t,j} := \left|a_{t,j}^{(l_{\mathrm{a}})}(x)\, r_{t,j}^{(l_{\mathrm{a}})}\right|, \quad M_t := \left(\sum_{j\notin S} |a_{t,j}^{(l_{\mathrm{a}})}(x)\, r_{t,j}^{(l_{\mathrm{a}})}|^2\right)^{1/2}.$$

By the first-order expansion of L_f,

$$L_f(\widetilde{\mathbf{H}}^{(t)} + \eta u) = L_f(\widetilde{\mathbf{H}}^{(t)}) + \eta\langle g_t, u\rangle + o(\eta),$$

hence

$$G_t = \eta\big(\langle g_t, u_t\rangle - \langle g_t, u_t^s\rangle\big) + o(\eta) \le c\,\eta\,\|g_t - g_t^s\| + o(\eta).$$

Since the surrogate replaces the full MLP expansion Z by its projected version $Z_s = P_S Z$, its Jacobian with respect to the attacked hidden state is correspondingly projected: $\frac{\partial Z_s}{\partial \widetilde{\mathbf{H}}} = P_S \frac{\partial Z}{\partial \widetilde{\mathbf{H}}}$. Let

$$r_t = \nabla_Z L_f\left(\widetilde{\mathbf{H}}^{(t)}\right), \qquad r_t^s = \nabla_{Z_s} L_s\left(\widetilde{\mathbf{H}}^{(t)}\right).$$

Under the surrogate approximation, we identify the surrogate MLP-path gradient with the projected full gradient, i.e. $r_t^s \approx P_S r_t$. Hence

$$g_t^s = \left(\frac{\partial Z_s}{\partial \widetilde{\mathbf{H}}}\right)^\top r_t^s \approx \left(\frac{\partial Z}{\partial \widetilde{\mathbf{H}}}\right)^\top P_S r_t,$$

and therefore

$$g_t - g_t^s \approx \left(\frac{\partial Z}{\partial \widetilde{\mathbf{H}}}\right)^\top (I - P_S) r_t.$$

Accordingly, we use the bound

$$\|g_t - g_t^s\| \le \left\|\frac{\partial Z}{\partial \widetilde{\mathbf{H}}}\right\| \cdot \|(I - P_S) r_t\| + \varepsilon_t,$$

where ε_t collects the surrogate mismatch induced by replacing r_t^s with $P_S r_t$. We work in the ideal projected-gradient case, i.e., $\varepsilon_t = 0$, which we include in the local regularity conditions of Theorem 3.

For $j \notin S$,

$$|r_{t,j}^{(l_{\mathrm{a}})}| = \frac{s_{t,j}}{|a_{t,j}^{(l_{\mathrm{a}})}(x)|} \le \frac{s_{t,j}}{m_0},$$

and thus

$$\|(I - P_S) r_t\|^2 = \sum_{j\notin S} \big(r_{t,j}^{(l_{\mathrm{a}})}\big)^2 \le \frac{1}{m_0^2} \sum_{j\notin S} s_{t,j}^2 = \frac{1}{m_0^2} M_t^2.$$

Hence

$$\|(I - P_S) r_t\| \le \frac{1}{m_0} M_t.$$

Applying Lemma 1,

$$\left\|\frac{\partial Z}{\partial \widetilde{\mathbf{H}}}\right\| \le L_\phi L_\sigma \|W_1\| \left(C_{\mathrm{attn}} \frac{B_Q B_K B_V}{\sqrt{d_m}} + B_V\right).$$

Therefore

$$G_t \le \frac{c\, L_\phi L_\sigma \|W_1\|}{m_0} \left(C_{\mathrm{attn}} \frac{B_Q B_K B_V}{\sqrt{d_m}} + B_V\right) \eta\, M_t + o(\eta).$$

Summing over $t \in [K]$, we obtain

$$\sum_{t=1}^K G_t \le \frac{c\, L_\phi L_\sigma \|W_1\|}{m_0} \left(C_{\mathrm{attn}} \frac{B_Q B_K B_V}{\sqrt{d_m}} + B_V\right) \eta \sum_{t=1}^K M_t + o(K\eta).$$

By a union bound, the estimate holds with probability at least $1 - 2Kp$. Denote

$$C_{\mathrm{surr}} = \frac{c\, L_\phi L_\sigma \|W_1\|}{m_0} \left(C_{\mathrm{attn}} \frac{B_Q B_K B_V}{\sqrt{d_m}} + B_V\right)$$

gives the final result. □

E.5 A Spatial Transfer Limitation of Last-Token ReFT

We further explain why a ReFT module trained only at the final token need not transfer optimally to earlier positions. The key point is that, at an earlier position, the reconstruction error naturally decomposes into two parts: one outside the learned edited subspace, and one inside that subspace.

Proposition 1 (Orthogonal decomposition of transfer error under single-token LAT)**.** *Let* $(\mathbf{R}^*, \mathbf{W}^*, \mathbf{b}^*)$ *be obtained by training ReFT only at the final token position* T*:*

$$(\mathbf{R}^*, \mathbf{W}^*, \mathbf{b}^*) \in \arg\min_{\mathbf{R},\mathbf{W},\mathbf{b}} \mathbb{E}_{\mathbf{h}_T}\left[\max_{\|\delta\|\le\varepsilon} \|\Phi_{\mathbf{R}}(\mathbf{h}_T+\delta)-\mathbf{h}_T\|^2\right].$$

For an earlier position $t < T$*, let* $\tilde{\mathbf{h}}_t = \mathbf{h}_t + \Delta_t$*. Then*

$$\mathbb{E}\Big[\|\Phi_{\mathbf{R}^*}(\tilde{\mathbf{h}}_t)-\mathbf{h}_t\|^2\Big] = \mathbb{E}\Big[\|(\mathbf{I}-\mathbf{R}^{*\top}\mathbf{R}^*)\Delta_t\|^2\Big] + \mathbb{E}\Big[\|\mathbf{R}^{*\top}(\mathbf{W}^*\tilde{\mathbf{h}}_t+\mathbf{b}^*-\mathbf{R}^*\mathbf{h}_t)\|^2\Big].$$

Proof. By definition of the ReFT operator,

$$\Phi_{\mathbf{R}^*}(\tilde{\mathbf{h}}_t) = (\mathbf{I}-\mathbf{R}^{*\top}\mathbf{R}^*)(\mathbf{h}_t+\Delta_t) + \mathbf{R}^{*\top}(\mathbf{W}^*\tilde{\mathbf{h}}_t+\mathbf{b}^*).$$

Subtracting $\mathbf{h}_t$ and using

$$\mathbf{h}_t = (\mathbf{I}-\mathbf{R}^{*\top}\mathbf{R}^*)\mathbf{h}_t + \mathbf{R}^{*\top}\mathbf{R}^*\mathbf{h}_t,$$

we obtain

$$\Phi_{\mathbf{R}^*}(\tilde{\mathbf{h}}_t)-\mathbf{h}_t = (\mathbf{I}-\mathbf{R}^{*\top}\mathbf{R}^*)\Delta_t + \mathbf{R}^{*\top}(\mathbf{W}^*\tilde{\mathbf{h}}_t+\mathbf{b}^*-\mathbf{R}^*\mathbf{h}_t).$$

The first term lies in the orthogonal complement of the edited subspace, while the second lies inside the edited subspace. Since these two subspaces are orthogonal,

$$\|\Phi_{\mathbf{R}^*}(\tilde{\mathbf{h}}_t)-\mathbf{h}_t\|^2 = \|(\mathbf{I}-\mathbf{R}^{*\top}\mathbf{R}^*)\Delta_t\|^2 + \|\mathbf{R}^{*\top}(\mathbf{W}^*\tilde{\mathbf{h}}_t+\mathbf{b}^*-\mathbf{R}^*\mathbf{h}_t)\|^2.$$

Taking expectations yields the stated decomposition. □

The proposition highlights two distinct sources of transfer error when a ReFT module trained only at position T is reused at an earlier position $t < T$. The first term on the right hand side measures perturbation components that lie outside the learned edited subspace and therefore cannot be removed by the reused low-rank intervention. The second term measures reconstruction mismatch within the edited subspace itself. Together, these terms show that even if the final-token ReFT module is effective at position T, its zero-shot reuse at earlier positions need not remain optimal. This provides additional support for applying the defense over a suffix window rather than only at the final token.

F Estimated Compute Calculations

F.1 Estimated FLOPs

We estimate adversarial-training compute using dense-equivalent Kaplan-style FLOPs. Let N denote the dense parameter count of the full model and D denote the number of processed tokens. Following standard scaling-law approximations (Kaplan et al., 2020), we estimate one dense forward pass as

$$C_{\text{fwd}} \approx 2ND,$$

and one dense forward-backward training pass with parameter gradient accumulation as

$$C_{\text{fwd+bwd}} \approx 6ND.$$

When a backward pass is used only to obtain gradients with respect to an input or latent perturbation, and no parameter gradients are accumulated, we charge

$$C_{\text{input-bwd}} \approx 2ND.$$

Thus, one forward pass plus one input-gradient backward pass costs approximately $4ND$.

In the main table, we report normalized FLOPs per 512-token training example. These estimates are intended to compare training-time computation across methods. Unless otherwise stated, we use dense model parameter counts even when the implementation uses quantization or parameter-efficient wrappers.

R2D2. R2D2 first generates an offline adversarial pool and then fine-tunes the full model on the resulting data. We therefore decompose the cost into a full-model fine-tuning term and an offline search term:

$$C_{\mathrm{R2D2}} = C_{\mathrm{train}} + C_{\mathrm{search}}.$$

The training term uses full-model fine-tuning,

$$C_{\mathrm{train}} \approx 6ND_{\mathrm{train}}.$$

For the offline attack-generation term, if generating an attacked example requires n_{fwd} forward passes and n_{bwd} input-gradient backward passes over D_{search} tokens, we charge

$$C_{\mathrm{search}} = (2n_{\mathrm{fwd}} + 2n_{\mathrm{bwd}})ND_{\mathrm{search}},$$

summed over all generated adversarial examples. Since this search is performed offline, we amortize it over normalized 512-token training examples:

$$C_{\mathrm{R2D2\ step}} = \frac{C_{\mathrm{train}} + C_{\mathrm{search}}}{D_{\mathrm{train}}/512}.$$

LAT. Each inner PGD step uses one forward pass and one backward pass that computes gradients only with respect to the latent perturbation and stops at the attack layer l_{a}; we charge the latter $2\rho_{l_{\mathrm{a}}}NT$, with $\rho_{l_{\mathrm{a}}}$ defined as for our method below. The outer update backpropagates both the clean and adversarial losses through the full model, costing $2 \times 4NT$. Including the clean and adversarial forward passes and one shape-capture forward pass, the per-step cost with k inner steps is

$$C_{\mathrm{LAT}} = 2(k+3)NT + 2k\rho_{l_{\mathrm{a}}}NT + 8NT.$$

CAT. CAT performs a k-step continuous PGD inner attack and updates a LoRA adapter. Each PGD step requires one dense forward pass and one input-gradient backward pass through the dense model, so the inner-search cost is

$$C_{\mathrm{search}} = 4kNT.$$

The subsequent training update uses one clean forward pass and one adversarial forward pass through the dense model. Although only the LoRA adapter parameters are updated, gradients must still be backpropagated through the frozen dense network to reach the adapter modules. We therefore estimate the training-update cost as

$$C_{\mathrm{train}} \approx 4NT + 4NT + 2N_{\mathrm{LoRA}}T = (8 + 2p_{\mathrm{LoRA}})NT,$$

where N_{LoRA} is the number of trainable LoRA parameters and $p_{\mathrm{LoRA}} = N_{\mathrm{LoRA}}/N$. In our runs, $p_{\mathrm{LoRA}} < 1\%$, so this term is numerically close to $8NT$. Thus,

$$C_{\mathrm{CAT}} \approx 4kNT + (8 + 2p_{\mathrm{LoRA}})NT \approx (4k+8)NT.$$

Ours. Our method uses a pruned surrogate for the inner attack search and updates only a small ReFT intervention on the target model. Let N_{s} denote the dense-equivalent surrogate parameter count and N_{tr} denote the number of trainable intervention parameters. If the ReFT intervention is applied at layer l_{r}, the latent attack perturbation is applied at layer l_{a}, and the model has L_{all} layers, define

$$\rho_{l_{\mathrm{r}}} = \frac{L_{\mathrm{all}} - l_{\mathrm{r}} - 1}{L_{\mathrm{all}}}, \qquad \rho_{l_{\mathrm{a}}} = \frac{L_{\mathrm{all}} - l_{\mathrm{a}} - 1}{L_{\mathrm{all}}}.$$

Here $\rho_{l_{\mathrm{r}}}$ and $\rho_{l_{\mathrm{a}}}$ approximate the fraction of layers traversed by backward computation after the intervention or attack layer.

The target-model update requires a clean forward pass, an adversarial forward pass, and a backward pass from the loss to the ReFT intervention. We estimate this cost as

$$C_{\mathrm{target}} \approx 4NT + 4\rho_{l_{\mathrm{r}}}NT + 4N_{\mathrm{tr}}T.$$

The final term accounts for gradient accumulation on the trainable intervention parameters and is typically negligible relative to dense model computation.

For the surrogate inner search, each PGD step uses a surrogate forward pass and a backward pass from the output to the latent attack layer. Therefore,

$$C_{\mathrm{surrogate}} \approx 2k(1 + \rho_{l_{\mathrm{a}}})N_{\mathrm{s}}T.$$

The total per-step cost of our method is

$$C_{\text{ours}} = C_{\text{target}} + C_{\text{surrogate}}.$$

For surrogate runs where the exact dense-equivalent surrogate parameter count is not directly available, we approximate

$$N_{\text{s}} \approx N\left(1 - f_{\text{MLP}} r_{\text{p}}\right),$$

where f_{MLP} is the fraction of dense model parameters in MLP blocks and r_{p} is the pruning ratio of MLP neurons.

F.2 Measured Compute Time and Peak Memory

We additionally report implementation-level timing of the dominant adversarial training computation and the peak GPU memory of the training process. All measurements use NVIDIA H200 NVL GPUs, bfloat16 precision, a maximum sequence length of 512, and eight inner PGD/search steps. The per-device batch size is 8 for Pythia-1.4B and 4 for Qwen2.5-3B. Timing is averaged over 100 measured steps for Pythia-1.4B and 50 measured steps for Qwen2.5-3B after excluding five warm-up steps.

Table 10: Measured implementation-level compute time and peak training memory under matched configurations. Timing follows the native training-computation region instrumented for each implementation.

Model	Method	Compute time/step (s)↓	Peak memory (GB)↓
Pythia-1.4B	CAT	0.7908	25.29
	LAT	0.6517	25.39
	LAT-ReFT	**0.4442**	**11.01**
Qwen2.5-3B	CAT	0.8613	51.18
	LAT	0.6469	56.44
	LAT-ReFT	**0.5650**	**14.77**

In these implementation-level measurements, the recorded compute time of LAT-ReFT is 31.8% lower than that of LAT on Pythia-1.4B, closely matching the 31.5% reduction in estimated FLOPs, and 12.7% lower on Qwen2.5-3B. LAT-ReFT also reduces peak GPU memory by 56.6% and 73.8%, respectively. On Qwen2.5-3B, the measured time reduction is smaller than the estimated FLOPs reduction, which we attribute to fixed and implementation-dependent costs, such as kernel launches, synchronization, and memory operations.